RULE LOGIC AND ITS VALIDATION FRAMEWORK OF MODEL VIEW DEFINITIONS FOR BUILDING INFORMATION MODELING

A Dissertation
Presented to
The Academic Faculty

By

Yong-Cheol Lee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Architecture

Georgia Institute of Technology
December, 2015

RULE LOGIC AND ITS VALIDATION FRAMEWORK OF MODEL VIEW DEFINITIONS FOR BUILDING INFORMATION MODELING

Approved by:

Professor Charles M. Eastman, Advisor
College of Architecture and Computing
Georgia Institute of Technology

Dr. Rafael Sacks
Faculty of Civil and environmental Engineering
Israel Institute of Technology

Dr. Russell T. Gentry
College of Architecture
Georgia Institute of Technology

Dr. John R. Haymaker
Director of Research
Perkins + Will

Dr. Yong K. Cho
College of Engineering
Georgia Institute of Technology

Date Approved: November 5, 2015

“Good research contributes to changing society”

Chuck Eastman

ACKNOWLEDGEMENTS

First and foremost, I appreciate all the support of my advisor, Professor Chuck Eastman, for seeing me through the arduous process of this dissertation. It has been a great honor and privilege for me to learn and work with him. I have witnessed that he is always passionate about learning new things and continuously contributes to society. His enthusiasm and insight as a scholar was a great motivation to me. I thank him for all his contributions of time, ideas, and support to make my Ph.D. experience instructive and stimulating. I also thank Mary Claire Eastman for her generosity toward me and my family. I am also grateful to the Digital Building Laboratory (DBL) for providing me with invaluable opportunities to use state-of-the-art information in my research and to interact with academic and industry professionals. This group, its members, and friends have contributed immensely to my Ph.D. life at Georgia Tech.

With regard to my dissertation, I thank all the committee members for providing insightful comments and suggestions for my dissertation. Dr. Russell Gentry, Dr. Yong Cho, Dr. Rafael Sacks, and Dr. John Haymaker took significant time and devoted great effort to improving the quality of this research and guiding my research in the right direction. With their valuable reviews and inspirational discussions regarding my dissertation, I believe that I will be able to extend and clarify my research in BIM interoperability. In addition, I am especially grateful to my colleagues, Donghoon Yang, Kereshmeh Afsari, Samaneh Zolfagharian, and Shani Sharif, all of whom worked on projects related to the model view definition (MVD) of the Precast Concrete Industry.

Specifically, I appreciate the collaboration with Tim Chipman for supporting the implementation of MVD validation and providing insightful ideas. Without his help, I could not have completed this dissertation.

I would like to acknowledge the help and guidance of Minjung Maing, who allowed me to initiate BIM research for my master's degree in the Building Construction Program at Georgia Tech. In addition, consistent advice I received from Kathy Roper has motivated me to conduct interdisciplinary studies for facility management and BIM. I'd also like to thank Drs. Chiho Seo and Kyunghwan Kim at the KonKuk University, who encouraged me to study in the United States and have continued to support me in my effort to become an academic researcher. I also thank Jeff Saunders at Autodesk for giving me the opportunity to learn state-of-the-art BIM technology and work at one of the best BIM companies at San Francisco. Support for my research from the PCI, the American Institute of Steel Corporation (AISC), Autodesk, and HOK have been immensely helpful, facilitating my research efforts during my dissertation work. In addition, the support of the Atlanta and San Diego chapters of the International Facility Management Association (IFMA) during the last five years gave me great strength and allowed me to move forward.

In addition, I acknowledge my dearest mentors at the DBL: Ghang Lee, Donghoon Yang, Jinkook Lee, Jaemin Lee, Frank Wang, and Manu Venugopal. Specifically, I thank Ghang Lee for his continuous encouragement and thoughtful guidance, Donghoon Yang for his academic advisement and friendship, and Jinkook Lee for ushering me into the

realm of BIM. Other past and present friends that I have had the pleasure to work with are graduate students Wawan Solihin, Mehdi Nourbakhsh, Marcelo Bernal, Paula Gomez, and Matthew Swarts. I would also like to enumerate my sincere friends: Junha Kim, Taelim Choi, Sanghoon Lee, Hoyoung Kim, Yoonsang Lee, Sangwoo Sung, Hyunbo Seo, Eunwha Yang, Yongsung Lee, Junghui Lee, Jihyun Kim, Jaeho Yoon, Jeongil Park, Yeonsuk Jeong, Manghoon Kang, Jonghoon Kim, Jeonghyun Aum, Seungyeon Choo, Sungil Ham, Yeonsook Heo, Seunghyun Lee, Heawon Jeon, Sangpil Lee, Minsoo Back, Lisa Lim, Sungjin Kim, Sungjun Kim, Jaesuk Park, Seyoung Oh, Chaehee Han, Kyungki Kim, Jiwoon Park, Pileun Kim, Youngmi Choi, Byungcheol Kim, Marisabel Marratt, Chen Feng, Pieter Pauwels, Sijie Jhang, and all others for their feedback and friendship. I cannot underestimate the help I have received from Jane Chisholm for improving my English writing and oral presentation skills. My time at Georgia Tech was enriched by the Georgia Tech Korean Student Association, the Korean Community Presbyterian Church, the Buckhead Church, Dokdo United, and the Volcano Tennis Club.

I am eternally grateful for all the love and encouragement of my family. I thank my parents for raising and supporting me in all my pursuits, my sisters for their continued encouragement, and my parents-in-law for praying for me during my studies. Last, I appreciate my loving, supportive, encouraging, and patient wife, Jiae Son, for her faithful support and dedication throughout all of the stages of my study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AEC/FM	Architecture, Engineering, Construction and Facility Management
ACI	American Concrete Industry
ADA	Americans with Disabilities Act
AISC	American Institute of Steel Construction
API	Application Programming Interface
BIM	Building Information Modeling
BCF	BIM Collaboration Format
BPMN	Business Process Model Notation
bSI	buildingSMART International
CAFM	Computer-Aided Facility Management
COBie	Construction Operations Building Information Exchange
CV	Coordination View
DBL	Digital Building Laboratory
EMs	Exchange Models
GTDS	Global Testing and Documentation Server
GUID	Globally Unique Identifier
GUI	Graphical User Interface
IAI	International Alliance for Interoperability
IDM	Information Delivery Manuals
IFC	Industry Foundation Classes
IfcSvr	IFC server
IFD	International Framework for Dictionaries

LOD	Level of Development
MVD	Model View Definition
NBIMS	National Building Information Modeling Standard (NBIMS)
O&M	Operation and Maintenance
PCI	Precast Concrete Institute
RPC	Remote Procedure Call
SDK	Software Development Kit
STEP	STandard for the Exchange of Product Model Data
UUID	Universal Unique Identifier

SUMMARY

With the growing number of complex requirements for building and facility projects, diverse domain experts iteratively exchange building design and product data during the design, construction, and facility management phases. Such data exchanges, however, frequently involve unintended geometric transformations, inaccurate project requirements, and insufficient syntactic and semantic elements in building model data. Such issues exacerbate the problem of model integrity and cause expensive modifications, particularly during the late project phases such as the construction and operation stages. In addition, since project participants employ diverse types of building information model authoring software, their product models cannot be readily and efficiently exchanged among domain professionals and subcontractors.

For formalized specifications of data exchange, experts in diverse disciplines have defined the Model View Definition (MVD), which encompasses sets of particular specifications of building information data depending on the exchanges that occur during the process of a practical building project. Even though IFC, one of popular neutral formats for data exchange, exists, most Building Information Modeling (BIM) tools execute heterogeneous binding processes in their IFC importer and exporter that map their native objects and attributes to IFC data. As a result, once an IFC instance file is translated according to the specifications of model views through the IFC interfaces of BIM authoring tools, end-users and software developers find the evaluation of the IFC instance file demanding with regard to conformance to the specifications of a targeted MVD.

To ensure the interoperability of building information models, this dissertation includes an examination of model view rules categorized from the Precast Concrete Institute (PCI) model views and a generalization of the rule logic and structures of each rule set. Moreover, rule logic is coded and implemented on modularized validation platforms of a validation tool referred to as the IfcDoc tool, an automated model view documentation and validation application. The modularized validation platforms employ an object-oriented checking method that addresses various types of rule sets of model views. In addition, this dissertation includes a thorough discussion of the complexity and challenges of model view rule checking.

This dissertation is expected to help domain experts evaluate whether building design data fulfill the data exchange specifications of their domain and the objectives of a proposed project. Furthermore, to identify unreliable and inconsistent IFC mapping procedures of BIM authoring tools, software developers using the proposed approach would implement an automated debugging process in their IFC interfaces according to the specifications of a targeted model view. Hence, the automated validation framework allows project participants and software vendors to confirm if received building model data and IFC translation processes comply with exchange specifications regarding syntax defined in the IFC schema and semantics pertaining to a corresponding MVD.

CHAPTER 1 INTRODUCTION

1.1 OVERVIEW OF BUILDING INFORMATION DATA EXCHANGE

With the growing number of client-defined specifications and regulatory standards, the requirements of a building project have increased in scope and complexity. The proper management of such requirements, fulfilled by various domain experts in the design, construction, and facility management industries (Ghang Lee 2006; Eastman, Teicholz et al. 2011) ensures the successful delivery of a project and the reliable performance of subsequent procedures (McGraw-Hill 2009). To fulfill such requirements of a building project, domain experts iteratively exchange building information models throughout the design and construction phases because a building project has the same goal that must be achieved by active collaboration of domain professionals. Exponentially grown data that should be exchanged among project participants applying such building models has made the exchange of building data extremely complicated. For instance, the precast concrete industry (PCI) identified 47 distinct data exchanges of a precast concrete building design, many of which are iterated multiple times from the concept design to fabrication phases (Eastman, Jeong et al. 2009).

During such exchange processes, building data must satisfy particular data exchange specifications of a domain industry. Such requirements, however, are often not satisfied when domain experts exchange building information data multiple times. Thus, data exchange of a building design may result in syntactic problems or programmatic errors in

its data that can compromise the integrity and the interoperability of a building information model, leading to considerable overhead (Eastman, Lee et al. 2015). In addition, because of the unreliable IFC interfaces of BIM software solutions, imported and exported BIM data through the IFC format are inconsistent (Howard and Bjo¨rk 2007; Olofsson, Lee et al. 2008; Eastman, Jeong et al. 2009). Thus, stakeholders need to capture and keep track of building data to ensure the interoperability of building models pertaining to omissions, geometric transformations, and semantic changes. Validation of BIM data according to such requirements demands well-organized checking specifications and implementable validation frameworks (Lee, Eastman et al. 2014). In other words, when project experts iteratively send and receive BIM data (Froese 2003), such data must be validated within the framework of a programmed and automated procedure in which a receiver ensures whether a sender's building data are met both syntactically and semantically and whether an edited building design conforms to the specifications of a targeted exchange process (Eastman, Jeong et al. 2009).

To define specifications for data exchanges, domain experts and stakeholders use a model view that encompasses the predefined syntactic and semantic requirements of IFC instance models. That is, a model view can consist of criteria used for evaluating IFC instance files according to the corresponding specifications of data exchange. For achieving the described goals, this dissertation proposes a new approach to generalizing and integrating types of validation rule logic associated with a model view and implementing rule types for MVD validation. A means of validating IFC instance files according to model views would provide software vendors with an opportunity to

evaluate their IFC interfaces such as import and export features according to the model views that they used for developing their IFC interfaces. This newly proposed approach, developed on top of the IfcDoc tool, employs the checking framework and libraries embedded in this application. To support these demands, this dissertation describes formalized rule types of MVD specifications and defines rule logic and checking frameworks that address various types of rules. It also includes a discussion and challenges of the complex structure of an MVD.

1.2 INDUSTRY FOUNDATION CLASSES AND MODEL VIEW DEFINITION

1.2.1 Background and Definition Structure of Model Views

As client-provided building projects have increased in complexity with numerous requirements, domain professionals use a neutral file format that can be exchanged among heterogeneous building information model authoring tools (Lee and Eastman 2015). The IFC schema is one of neutral file formats that architecture, engineering, construction and facility management (AEC/FM) industries have broadly employed. The specifications of the IFC schema defined in the EXPRESS language, encompass diverse modeling constructs, data exchange definitions, and general relational rules. Since data exchanges during the design, construction, and facility management phases may need distinct sets of building design data, each domain of expertise requires one or more IFC MVD representing particular specifications (Lee and Eastman 2015). In addition, IFC is used based on the requirements of the MVD (IAI 2003; Sacks, Kaner et al. 2010). The MVD also includes the exchange processes of specific project data and the specifications of applicable data exchanges among BIM authoring tools (Hietanen 2006), facilitating

seamless data exchanges. Figure 1 shows the structure of an MVD and illustrates reuses of concepts. The implementation agreement of a concept contains an IFC mapping for native objects that is helpful for software vendors to develop their IFC interfaces.

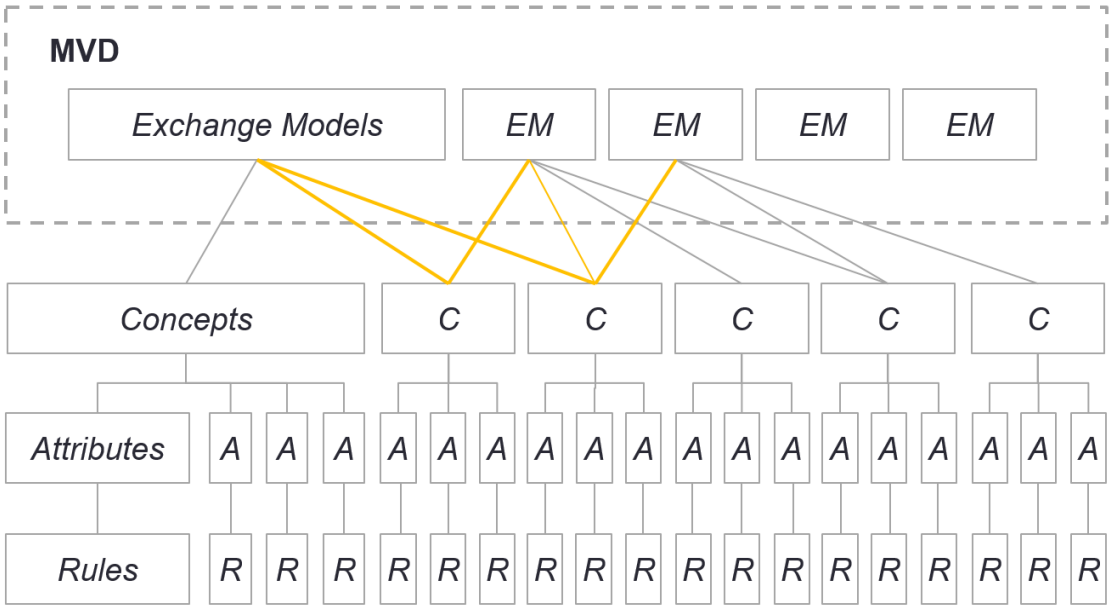


Figure 1 Structure of a model view

A set of concept descriptions modularize domain knowledge so that predefined modules can be reused for organizing model views of other domains. In other words, the structure of an MVD employs the modularized descriptions called concepts (See 2014). To avoid redundancy of specifications, modularized concepts are used to define the high-level or low-level requirements of an entity, a relationship, and an attribute as model views are specified. Each mapping and implementation agreement is defined in an attribute of a concept. A concept contains multi-attributes that can efficiently signify binding requirements of associated IFC entities, relationships, and properties: Thus, definitions

of attributes can be regarded as one rule that a corresponding entity must fulfill. In addition, according to its rule, an IFC interface translates its native objects.

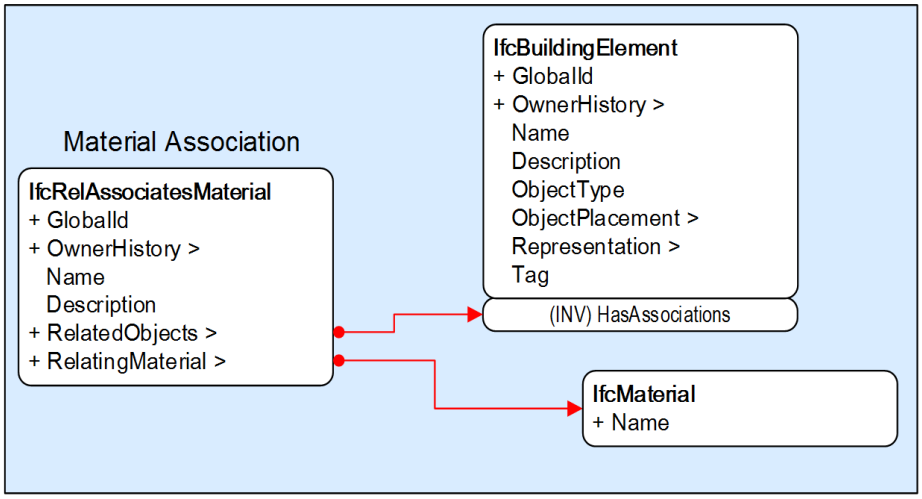
The reusability of concepts is a primary benefit in defining model views because an automated validation for one concept can be reused iteratively. With regard to reusability of a concept, two distinct levels of a concept reuse exist: reuse by domain industries and reuse by concepts that have a supertype entity as a root entity. First, a concept that can have multiple rules is a unit of the specifications of data exchanges. Since a concept is a modularized document, it is subject to be reused by several exchange models and shared with diverse domains. As shown in Figure 1, several exchange models can reuse defined concepts for describing exchange requirements of diverse domains. In case of concept reuse by several domains, a concept is used again as itself without any changes and updates.

Another case is the reuse of a concept by other concepts. A concept is a modularized unit of knowledge that can be reused by several concepts having a subtype entity as a root entity. For example, Figure 2 is a concept description specifying the requirements for the precast piece material association. The concept requires that IfcBuildingElement should have the reference of the IfcRelAssociatesMaterial entity to specify its material association (Lee and Eastman 2015). This concept has several attributes: GlobalId, OwnerHistory, RelatedObjects, and RelatingMaterial attributes. To represent a material relationship, this concept is subject to be reused by several subtypes of IfcBuildingElement such as IfcWall and IfcColumn. Since the specifications of

implementation agreements of concepts have requirements for attributes, this dissertation focuses on them to generalize rules for model view checking. Since each concept encompasses implementation specifications of attributes and relationships, the objective of this dissertation is to make them implementable as checking criteria to validate IFC instance files. A concept entails implementation agreements that describe associated attributes and their requirements defined for data exchanges. The MVD composes certain types of concepts to represent specifications of a particular exchange process.

IFC Release Specific Concept Description (IFC 2x3)					
Precast Piece Material Association					
Reference	PCI-061	Version	1.1	Status	Draft
Relationships	Assigns material to either precast or non-precast elements.				
History	Developed Fall, 2009, revised for submission November, 2012				
Authors	Ivan Panushev, Chuck Eastman (chuck.eastman@coa.gatech.edu)				
Document Owner	Precast/Prestressed Concrete Institute				

Instantiation diagram



Implementation agreements

Attribute	Implementation agreements
GlobalId	Must be provided
OwnerHistory	Must be provided, but may contain dummy data
Name	Optional
Description	Optional
RelatedObjects	Must be subtype of IfcBuildingElement.
RelatingMaterial	Must be the IfcMaterial

Figure 2 A concept document defined for the precast piece material association

1.2.2 The MVD Development Process of the National Building Information Modeling Standard (NBIMS)

Domain experts and software vendors define the requirements for data exchanges in the format of the IDM (Eastman, Sacks et al. 2009). The IDM consists of a process model that is usually drawn in Business Process Model Notation (BPMN), which is a graphical representation for describing business processes and data exchange flows (White 2004). Such specifications in the IDM are translated by MVD developers into the IFC format in order to generate IFC sub-schema for particular domain knowledge (Eastman, Jeong et al. 2009). This IFC MVD, which encompasses data exchange specifications and requirements for end-users and IFC implementation (Hietanen 2006), helps ensure interoperability and accuracy of IFC data exchange implementation pertaining to predefined domain specifications. As discussed in the previous section, to efficiently organize the implementation requirements iteratively used by several entities, exchange models of the MVD are generated by applying modularized documents referred to as concepts. Such concepts are consumed by software vendors to develop IFC interfaces and also used in validating them. Figure 3 represents a data flow and processes of model view definition.

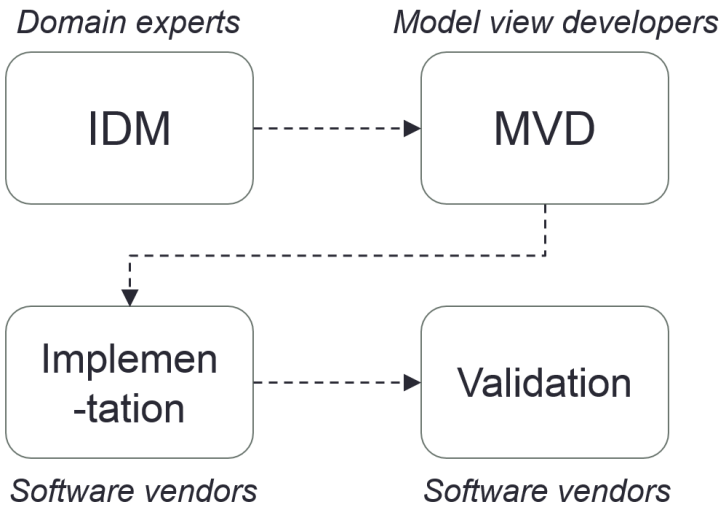


Figure 3 Simplified model view definition flow

The NBIMS contains the definition processes of building information data exchanges for leveraging interoperable business contexts (buildingSMART 2007). Figure 4 shows the development process of a model view defined in the NBIM: it contains phases for an information exchange template, an IDM, an MVD, implementation, validation, and deployment. The information exchange template provides functional and semantic requirements of all BIM data exchanges for a particular domain (Lee, Eastman et al. 2015).

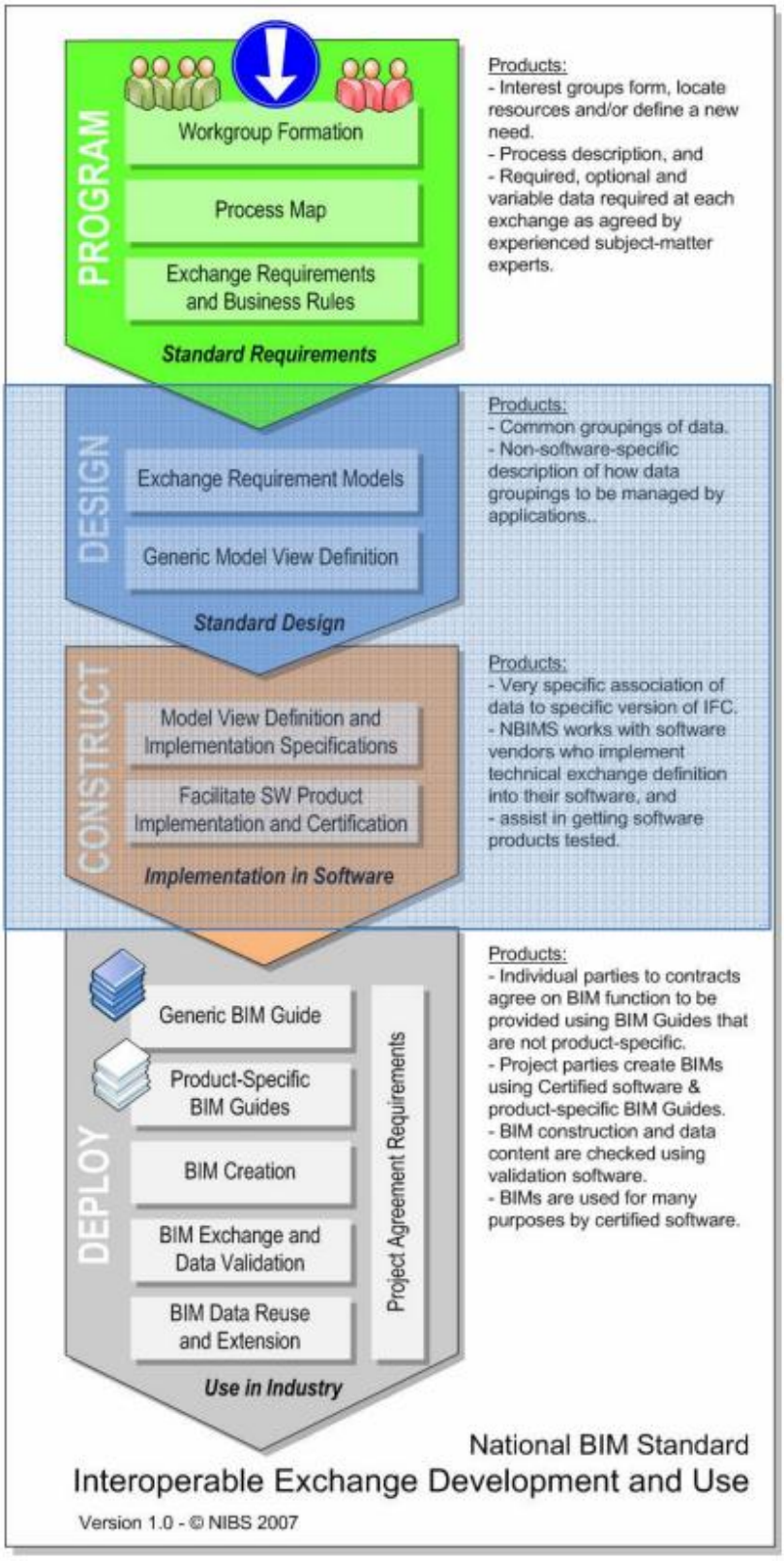


Figure 4 The MVD development process of the NBIMS

Figure 5 shows the IDM for a slab system documented in an EXCEL spreadsheet. The IDM should include the accurate descriptions of these data exchange requirements to be translated into specifications subject to be used by software vendors (Eastman, Jeong et al. 2009). The translated specifications defined in the IFC schema is an MVD that embeds project requirements and exchange rules that are practical and regulative components imperative for low-level goals of each domain. Implementation agreements defined in an MVD help software vendors develop consistent IFC interfaces and facilitate BIM data exchanges among industry domains (Eastman, Jeong et al. 2009).

Information Items	Attribute Set	Attributes		A_EM.1
Slab Systems, Building Cores	Geometry	Extruded shapes or solid forms	Required?	R
			Function?	V
			Accuracy?	P
	Material	Material type	Required?	R
		Quantity	Required?	
	Finishes	Geometry	Required?	O
			Function?	V
			Accuracy?	P
		Surface treatments	Required?	O
	Assembly relations	Part of building	Required?	
	Association relations	Implements structrual object..	Required?	
	Nested relations	Contains components	Required?	
	Connection relations	.. to Precast	Required?	
		.. to CIP	Required?	
		.. to Steel	Required?	
	Meta Data	Author, Version, Date	Required?	O
		Approval Status, Date	Required?	O

Figure 5 Information Delivery Manual for a Slab System

1.3 MOTIVATION AND PROBLEMS IN CURRENT PRACTICE

In the BIM handbook, Interoperability is defined as a capability to exchange BIM data between its authoring tools, facilitating workflows (Eastman, Teicholz et al. 2011). Since several project participants and stakeholders participate in a project, an interoperable data exchange is tremendously critical in achieving the goal of a project. All the distinct entities have diverse nomenclatures, geometries, data formats and schemas, accuracy, and level of development (LOD). To resolve these issues, numerous efforts have been made to leverage standardization and interoperability for the AEC/FM domains during the last decades (buildingSMARTInternational 2015).

Various domain experts have defined the MVD translated from the IDM (Eastman, Jeong et al. 2009). Documented MVD references for developing IFC interfaces of their BIM authoring tools allow software vendors to translate native models in the IFC ones. While defining exchange specifications needed for each exchange of building design data, since such specifications include paper-based documents or inconsistently configured structures, the use of the MVD is somehow limited by particular software vendors. In addition, these limitations are dependent on semantic insufficiency of data models and on misinterpretations and inaccurate coding of needed specifications (Eastman 2015).
In particular, after developing IFC interfaces, if they have IFC binding errors leading to unconformity with regard to the MVD specifications, such problems are not easily identified, resulting in that software vendors have to debug numerous binding classes of IFC import and export features. In addition, because the MVD consists of data exchange specifications, domain professionals can use them as criteria for evaluating IFC files when exchanging them in each process. Despite these practical needs and benefits, the MVD is still a paper-documented or inactive set of specifications. Another challenge is inconsistency of model view requirements. Because of the lack of an MVD standard, use cases include various types and terminology of business rules that might declare inconsistent requirements not supposed to be handled in the MVD.

As a result of these challenges, validation of an IFC instance file has been recognized critically. Because an inconsistent MVD encompasses the vague scope of rule-sets, domain professionals need to evaluate building instance files pertaining to their own MVD according to corresponding exchanges. The MVD is expected to provide an

explicit binding document so that software venders can develop IFC interfaces on their native applications. Unstructured specifications and unorganized implementation rules, however, can cause confusion in defining MVDs and executing predefined rules. Moreover, a debugging process that identifies the causes of errors in transforming native objects to and from IFC instances entails time-consuming and repetitive tasks. If an IFC instance file exported from a BIM authoring tool does not meet the requirements of the MVD and the IFC specifications, the building model data results in serious syntactic and semantic problems that can cause translation errors, omissions, and failures to read a file. To address these critical issues, types of the MVD specifications and associated formal rule logic must be identified and defined explicitly. An integrated validation framework of MVDs with an implementable binding process must also be developed for ensuring interoperability and facilitating a mapping process on native modeling applications. This validation process not only helps the MVD be consistent pertaining to definitions and specifications of building objects defined in the diverse terms, but also makes the best use of the defined MVD because the MVD is typically not used frequently after software vendors use them for developing IFC interfaces (Lee, Eastman et al. 2015).

Figure 6 represents the attributes and the relationships of the IfcSlab entity defined in the PCI MVD. The diagram illustrates how entities and attributes are connected and what values and types are required for them. The IfcSlab entity has two types of attributes: explicit and inverse attributes. In Figure 6, the explicit attribute exists in an ISO 10303 Part 21 physical file (P21 file) and the inverse attribute never occur in IFC files. The explicit attributes such as GlobalId, OwnerHistory, and Name are black-colored and the

inverse ones such as HasAssignments, IsDecomposedBy, and Decomposes are gray-colored in the diagram. The black line represents the semantic link: for example, GlobalId must have type of IfcGloballyUniqueId and ObjectPlacement must refer IfcObjectPlacement. In case of the ObjectType attribute, the relationship with the IfcLabel describes that ObjectType must have one of IfcLabel values, Slab and Precast Slab. These relational skeleton and semantic requirements are defined in the implementation agreement of a concept. Based on the relevant concept, the diagram depicts that the IfcSlab entity of a P21 file must include attributes for GlobalId, OwnerHistory, ObjectType, ObjectPlacement, Representation, and PredefinedType. In addition, attributes with no relationship such as Name, Description, and Tag describe that the IfcSlab optionally contains values and types for the attributes. These specifications are supposed to apply to a P21 file exported from BIM authoring tools. Thus, these requirements are expected to be satisfied by a P21 file while exchanging BIM data among domain experts through various product data authoring tools.

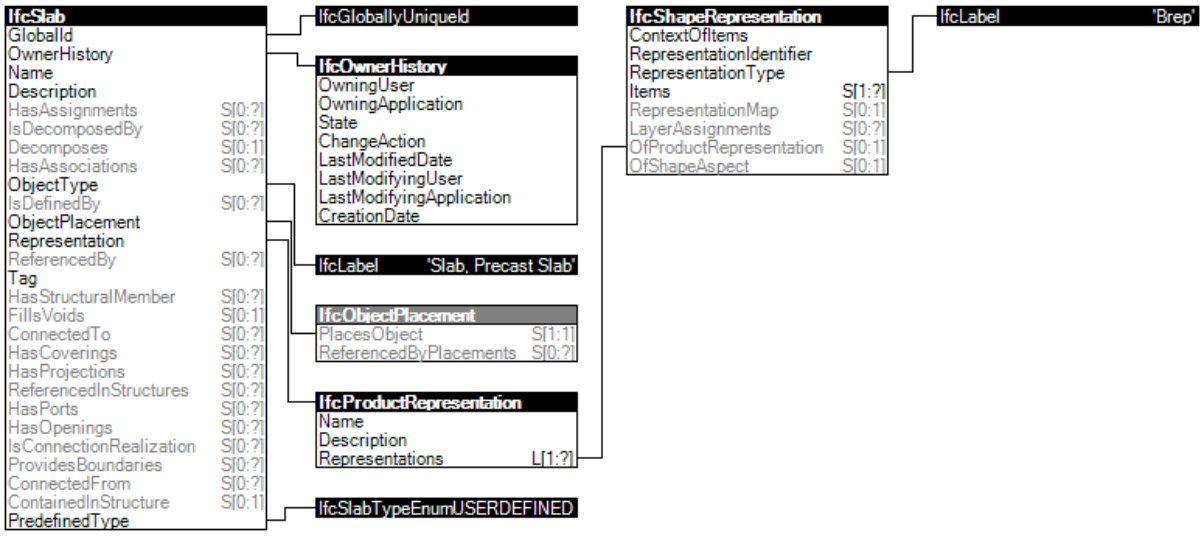


Figure 6 The structural relation of entities and attributes of IfcSlab defined in the PCI-153 concept, precast slab attributes

Figure 7 illustrates how the implementation agreement can be applied to instances of a P21 file. The figure consists of the sample IFC model, the implementation agreement, and p21 file instances. The three instances, #111, #188, and #237, are describing the IfcSlab entity of a sample model, but have various sets of attributes. Essentially, the definition of an IfcSlab entity follows the IFC schema. Within the limits of a defined structure in the IFC schema, the implementation agreement specifies the requirements and constraints of the IfcSlab entity. In Figure 7, #237 IfcSlab instance complies with the requirements of the implement agreement. However, #118 IfcSlab instance has two conflicts in ObjectType and Representation attributes. Even though the implementation agreement specifies that the IfcSlab must have the "Precast Slab" value for the ObjectType attribute and must be referring to IfcProductRepresentation for the Representation attribute, #188 instance encompasses the Beam as ObjectType and null for Representation. If a P21 file does not satisfy these predefined requirements in the MVD, the P21 file has syntactic and semantic problems that cause translation errors, omissions, and fails to read a file.

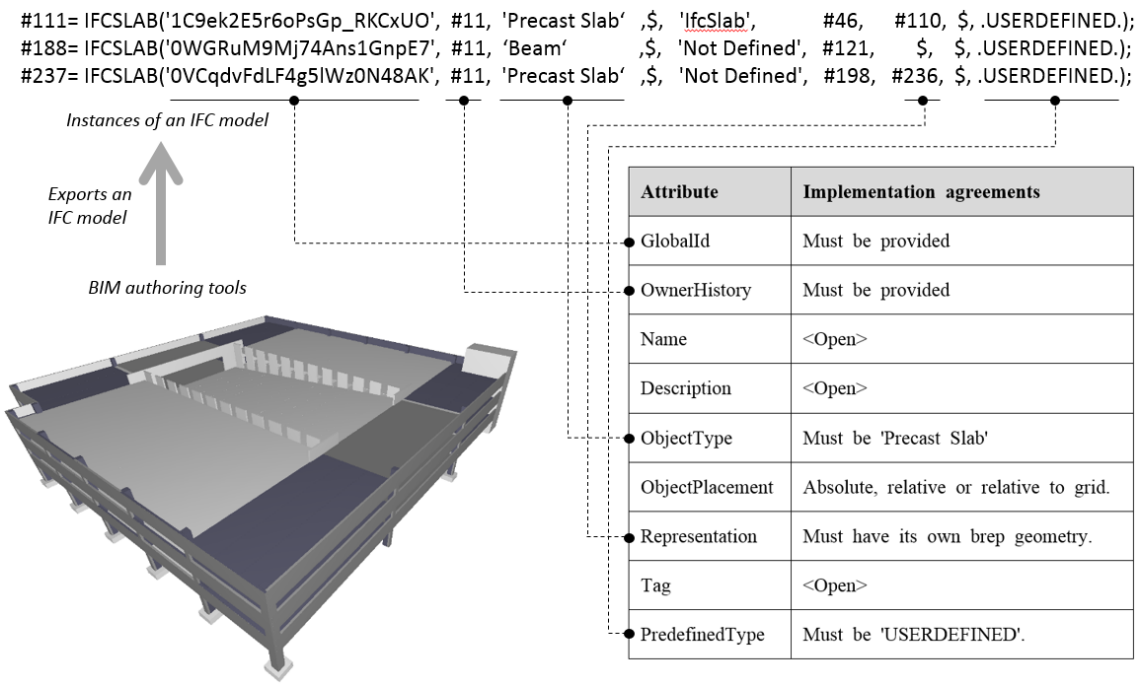


Figure 7 Validation of instances of an IFC model with regard to the implementation agreements of the PCI-153 concept, precast slab attributes

The error regarding the case of the #118 instance in Figure 8 is generally caused by applying the wrong binding structures of BIM authoring tools. Because the specifications of the IFC schema is open to vague interpretation (Fowler 1995), each BIM authoring tool has heterogeneous binding processes and consists of its own import and export features (Olofsson, Lee et al. 2008; Eastman, Lee et al. 2009), including heterogeneous mapping procedures that bind their native models into the IFC format. Furthermore, each BIM software generally supports heterogeneous subsets of the IFC schema descriptions applicable to its own software solution (Fowler 1995). For example, if a flat concrete slab of a native model shown in Figure 8 is exported to the IFC format from one BIM

authoring tool that has a binding process translating the object into IfcPlate, the IFC model contains a plate made of concrete instead of a concrete slab.

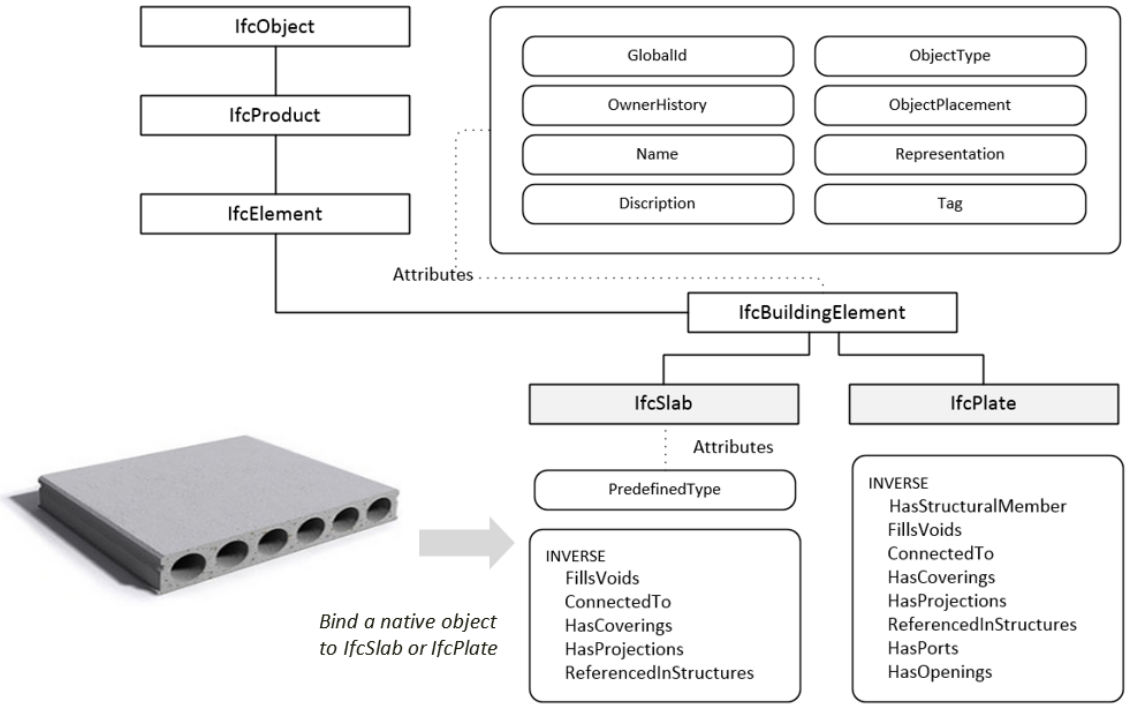


Figure 8 IFC schema hierarchy, attributes, and inverse relationships of IfcBuildingElement, IfcSlab, and IfcPlate

Another BIM authoring tool may map a concrete slab into an IfcSlab. Even though two entity types, IfcPlate and IfcSlab, share the attributes inherited from IfcBuildingElement that is a super type of two entities, IfcSlab has an attribute, PredefinedType, which IfcPlate does not contain. Moreover, they consist of various types of inverse relationships as shown in Figure 8. Since a concept definition specifies the requirements of one entity and its relationship, the required attributes and values of a slab defined by two pertinent IFC entities will differ from the semantic and syntactic perspectives based on the concept

definitions of both entities. These heterogeneous binding processes that can be caused by a developer of an IFC interface of BIM authoring tools might cause the problem in interoperability. Hence, if BIM authoring tools have binding processes that do not consider MVDs specification, users may receive unnecessary information and lack required data. To ensure that the MVD instance file is accurate, validation of the semantic and syntactic compliance of a P21 file is a prerequisite.

1.4 THE OBJECTIVES AND SCOPE OF RESEARCH

This dissertation started with a question: How can the definition and validation processes for model views of each domain industry be improved using formalized rule logic and a concrete validation framework more consistently than the current practice? Two major research questions raised in this dissertation and investigated are as follows:

(1) What types of checking logic and structure are required for MVD validation?

The information and knowledge of MVDs are too heterogeneous and unorganized to define the requirements of IDM and validate IFC instance files. In particular, the IDM pertaining to applicability to MVDs and a binding process for IFC interfaces has no concrete standard and baseline for defining exchange requirements. Furthermore, validating IFC interfaces are implemented by a method that the public cannot easily define and modify the rule sets. The main question is how MVDs and their rules can be integrated with native binding process. For this issue, explicit knowledge and formal specifications of MVD rules are required from the initial phase of defining scope and business rules for MVDs. This dissertation proposes a new approach to deal with MVD

validation using concept rule logic of the IfcDoc application. Rule logic is a generalized form of the MVD requirements retrieved from PCI concepts.

(2) What is a more integrated approach and method of checking capabilities to validate an IFC interface with regard to conformance to MVDs?

Current practice in defining a number of concepts and rules is also inefficient and not intuitive. In particular, lack of reusability of concepts and rules causes redundant information and potentially inconsistent information of current MVDs requirements. The IfcDoc tool is expected to apply rule logic in order to execute the rule sets of concept attributes efficiently. Based on the various types of rule implementation, rule logic is combined and recursive to address diverse requirements of the MVD. This dissertation also aims at providing an MVD validation interface so that the public can easily utilize and modify concepts and rules. In addition, to facilitating the MVD and rule definition process, rule logic would be used to find out existing concepts that have similar rule sets, which improve reusability of present concepts.

The ultimate objective of this dissertation is to define formal logic and algorithms of rule types with regard of the MVD and is to provide the concrete platform, which supports validating IFC instance files against the MVD rule types. Logically organized rules in accordance with concepts of the PCI MVD allow users to consistently validate P21 instance models pertaining to data exchange specifications. In addition, one of goals of this dissertation is to implement rule logic at the application level and to evaluate accuracy and checking capability of rule logic and structure. Furthermore, this

dissertation addresses the efficient methods that define the MVD rules on the IfcDoc application. The approach of rule definition would deal with translating an implementation agreement defined in the MVD into the concepts of the IfcDoc application and would include the ways to reuse the existing concepts. Since the validation process would be used to verify a binding process of software applications, when defining rule sets and composing them, checking features for MVDs rules should consider a structural unit of a binding process.

As the scope of this dissertation, the rule logic is generated and retrieved from 96 concepts of the PCI MVDs which consist of syntactic constraints pertaining to the IFC schema defined by the EXPRESS language and semantic rules with regard to business specifications. The total rules are approximately 480 on the assumption that each concept description entails five rules on average. Even though this dissertation suggests several rule logic identified from PCI MVDs, because model view is defined by domain experts' requirements within the scope of IFC schema, the rule logic might not cover exceptional rule checking of other domains. The evaluation of proposed rule logic is illustrated in Section 5 with defining rules existed in IFC Coordination View V2.0 and COBie. The additional rules that the rule logic cannot cover will be added in the extended version of rule logic. The generalized rule logic and structure of the PCI MVDs can formalize the MVD knowledge, help ensure interoperability of building information models, and reduce effort and time in evaluating instance files according to data exchange specifications. In addition, the proposed validation framework and the new MVD checking approach using the IfcDoc tool, which embeds diverse validation features with

regard to types of the MVD rule logic, enables users to define concept rules directly for their purposes and to validate a P21 file against the defined rules.

CHAPTER 2 BACKGROUND

2.1 OVERVIEW

Data translation and transition among diverse project participants throughout the design, construction, and operation phases often leads to insufficient integration and data loss (Autodesk Inc 2008). To resolve such problems of interoperability, the IFC schema has been accepted as the building industry standard for interoperability (Venugopal, Eastman et al. 2014). This neutral format provides a consistent syntax so that diverse domain professionals and stakeholders can iteratively exchange IFC instance files that has the same structure and modeling representations. However, even though they use the IFC file, they still have several problems in exchange design data regarding syntactic, semantic, and design programming requirements (McGraw-Hill 2009; Sacks, Kaner et al. 2010; Lee, Eastman et al. 2015; Lee, Eastman et al. 2015).

Since the IFC schema is open to diverse interpretations and modeling methods, domain experts defines model views called MVD. The MVD consists of a set of concepts (Hietanen 2006). A concept, a modularized part of a model view, is used by several entities that help avoid the redundancy of definitions of data exchange requirements. In particular, the concept can be reused by various industry domains (Lee, Eastman et al. 2015). Thus, MVD is presented by aggregated concept specifications of required entities and relationships. This section outlines the diverse problems of data exchanges using MVD and BIM models.

2.2 INTEROPERABILITY ISSUES

BIM model generally includes three requirements: a syntax, a semantic, and a design requirements (Lee, Eastman et al. 2015). This dissertation addresses a syntax and semantic validation issues. First, since the IFC schema is defined in the EXPRESS language (ISO 10303-11), which specify the data specifications of STEP (STandard for the Exchange of Product Model Data), an IFC instance file must follow its syntactic requirements. Figure 9 shows the syntactic requirements of the IfcProject entity defined in the IFC 2x3 schema. Since the IfcProject entity is one subtype of the IfcObject entity, it must exist in the schema definitions. The figure represents formal EXPRESS propositions pertaining to GLOBAL, UNIQUE, and WHERE rules. These rules are imperative to avoid the translation errors of BIM authoring tools.

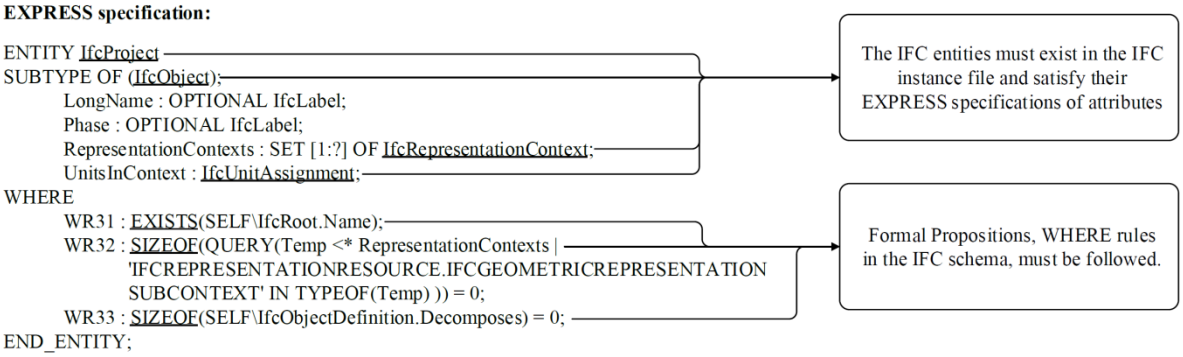


Figure 9 The IFC 2x3 schema definitions about the syntactic requirements of IfcProject (Lee, Eastman et al. 2015)

In terms of types of relationships of entities, an IFC instance file must fulfill the rules of the IFC schema (Hietanen 2006). As Figure 10 shows the requirements of IfcProject, an IFC instance file must have attributes of GlobalId, OwnerHistory,

RepresentationContexts, and UnitsInContext. To represent other necessary information for IfcProject, an IFC instance file can optionally contain values for attributes of Name, Description, ObjectType, LongName, and Phase (Lee, Eastman et al. 2015). In addition, IfcProject can be inversely referred by several relationships for HasAssignments, IsDecomposedBy, Decomposes, HasAssociations, and IsDefinedBy.

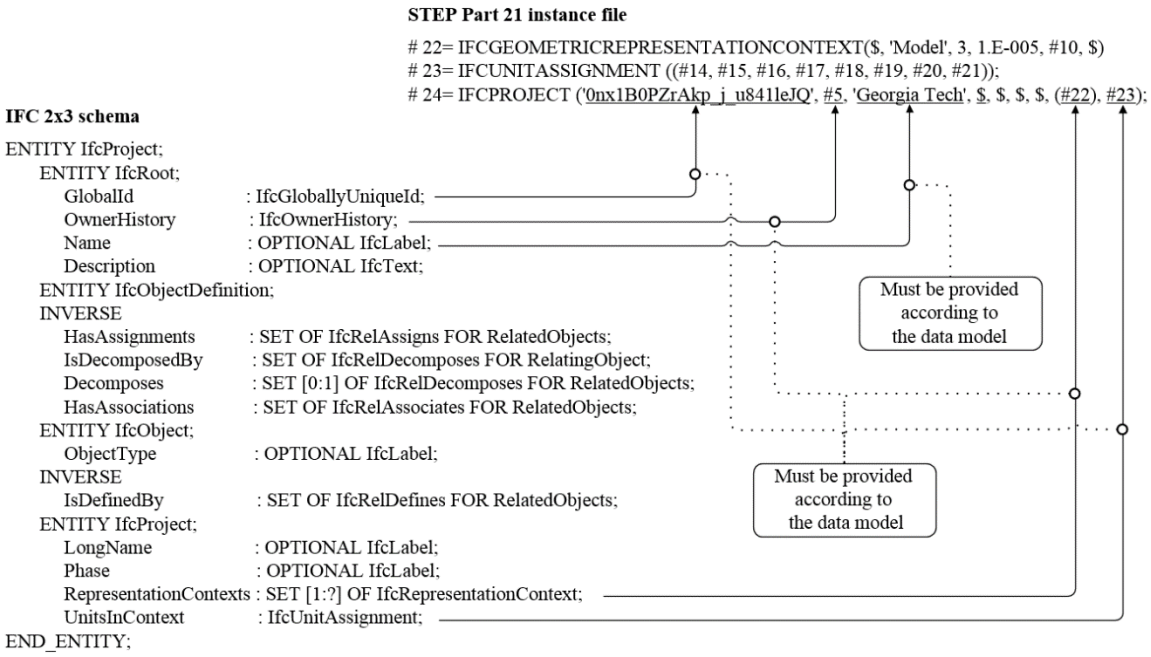


Figure 10 The required attributes of IfcProject defined in the IFC schema (Lee, Eastman et al. 2015)

With regard to semantic issues of interoperability, an IFC instance file should satisfy the semantic requirements defined in a corresponding MVD. Because an MVD declares the necessary specifications of data exchanges for a particular domain (Hietanen 2006; Eastman, Jeong et al. 2009), an IFC instance file used in such a domain should include required design information and relationships as defined in an MVD. Thus, an IFC interface such as IFC importer and exporter of BIM authoring tools should have an

accurate binding process that can translate requirements and verify the missing components. But, even though BIM authoring tools employ a same model view such as the IFC coordination view, they have heterogeneous IFC translation processes. These heterogeneous translations can generate IFC instance files that involve unintended geometric transformation and inaccurate project specificatiosn. Thus, based on an MVD, software vendors appropriately encode required specifications of data exchange into their IFC interface (Eastman, Lee et al. 2009).

2.3 RELATED EFFORTS AND RESEARCH

During an exchange of an IFC file between BIM authoring tools, the file typically goes through two translation steps: exporting and importing. These translators are developed based on model views that encompass the specifications of design exchanges of a particular domain. One model view broadly used in AEC/FM industries is the Coordination View discussed in Chapter 2. Thus, to ensure interoperability of a building model, a BIM data should be assessed according the specifications of an MVD. To meet the demand for accurate interoperability, researchers have devoted effort to studying the MVD and its validation.

LEE at el (Lee, Eastman et al. 2015), describes current six available applications for evaluating BIM data and presents their strengths and weaknesses: The Express Engine's EXPRESSO, the JDSAITM, the EXPRESS Data Manager™, the IFC server ActiveX Component, the IfcDoc, and the Solibri Model Checker® . This survey has an objective that identifies available software supporting validation of BIM data and structures three

processes of the validation: syntax, semantics, and design requirement checking. Two MVD validation approaches, IFC server-based checking and the IfcDoc tool, were surveyed in one study (Lee, Eastman et al. 2015). The strengths and weaknesses of such methods are illustrated in the paper, pointing out their insufficient checking features and limited validation scope.

Pertaining to validation of IFC interface of BIM authoring tool and IFC instance files of domain experts according to specifications of an MVD, no robust approach and platform support this checking process. Even though several industry domains, organizations, and governments have defined model views for their purposes, they do not have an approach to evaluating syntax and semantics of IFC instance files against the defined specifications of data exchanges. A number of errors in exchanging and translating IFC files results in the problems of BIM data exchange, making people have a negative perspective to using a neutral format. Since these interoperable problems are generally caused by inaccurate IFC binding assignments of BIM authoring tools, software developers have to manually identify human errors, technical problems, and other mapping errors of their IFC interfaces (Lee, Eastman et al. 2015). In addition, domain experts evaluate their IFC instance files to find out errors that cause unintended geometric transformations and incorrect data information. In order to improve these time-consuming and arduous debugging processes, automated validation of IFC instance files according to specifications of an MVD should be required (Weise, Liebich et al. 2009). Even though there is no robust approach to evaluating various checking scenarios of MVD validation,

there have been several effort for checking an IFC instance file according to diverse requirements such as the IFC schema and a building code (Solihin, Eastman et al. 2015).

With regard to IFC syntax checking, several commercial tools support such validation according to the EXPRESS language. The Express Engine, a light-weight validation tool, can evaluate an IFC instance file about the compliance of the EXPRESS language. Users also implement this tool for checking the IFC schema itself according to the EXPRESS language or a given schema (Lee, Eastman et al. 2015). In addition, the EXPRESS Data Manager™ one of commercial platforms that provides an object-based application for schema validation and object model development. This tool has a concrete checking platform that evaluate the syntactic requirements of the IFC schema and data files (Eastman, Lee et al. 2009). That is, these syntax checking applications cannot evaluate an IFC instance file with regard to the semantic requirements of data exchange processes (Lee, Eastman et al. 2015). The JSDAI™ is a library that helps reading and writing a data set specified in the EXPRESS language (Lee, Eastman et al. 2015). This tool provides a service evaluating the IFC schema according to given criteria such as the EXPRESS language.

For semantic validation of an IFC instance file against an MVD, buildingSMART international provides a service called the Global Testing and Documentation Server (GTDS). Figure 11 represents the web-documentation platform of GTDS that can implements validation of an IFC instance file IFC according to the specifications of the Coordination View 2.0 (CV 2.0) for software companies (buildingSMART 2010). This

service allows software vendors to evaluate their IFC interface and obtain a certification issued by bSI. However, because it is a web-based checking tool, users cannot identify and edit the requirements of concept documents. Hence, the rules and checking processes of an MVD cannot be reused by other industry domains. In particular, GTDS only support validation of an IFC interface according to CV V2.0 (Lee, Eastman et al. 2015). In addition, the evaluation process takes considerable time and requires certification fees for the approval of MVDs.

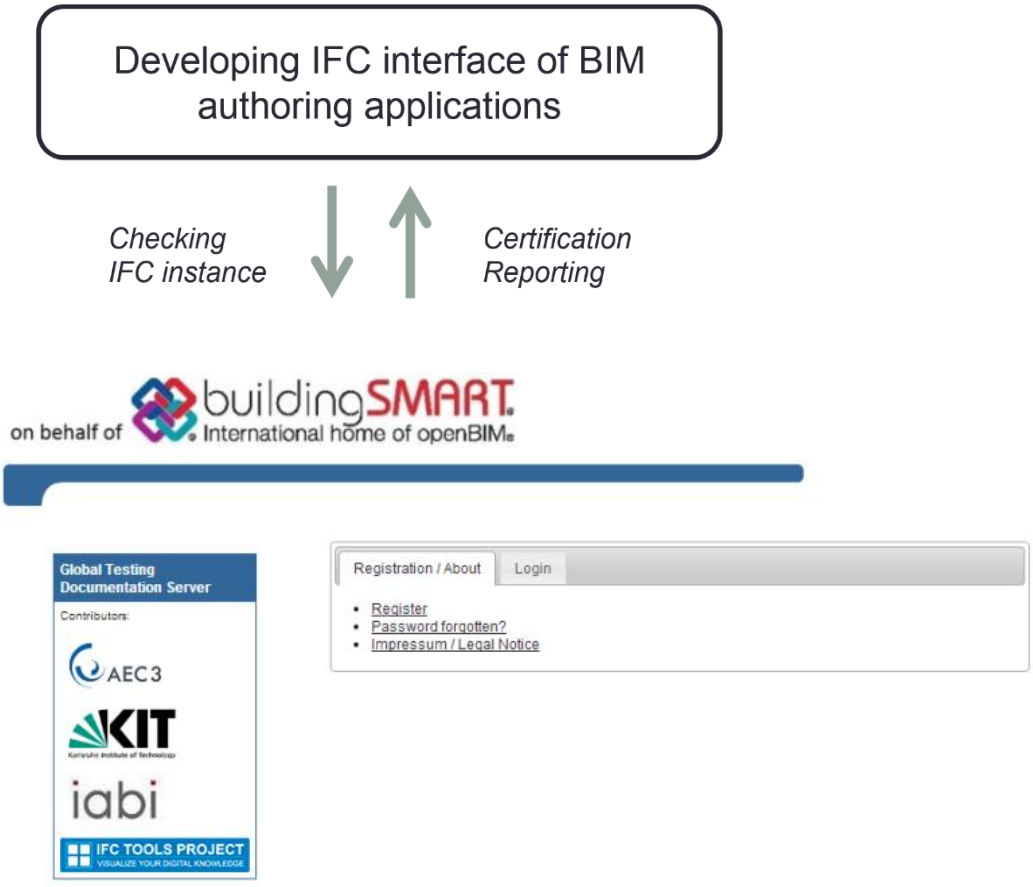


Figure 11 Global Testing and Documentation Server

IfcSvr-based checking using IFC server ActiveX Component (IfcSvr) is one validation platform that this dissertation used at the initial phase of the research. To resolve time-consuming and insufficient validation issues, the Digital Building Laboratory at Georgia Tech developed an object-based validation method that executes modularized rule checking using the testing libraries of the Digital Alchemy (Lee, Eastman et al. 2015). The IfcSvr contains the data sets of the IFC schema and capabilities of referential evaluations (SECOM and VIT 2001). For example, users can retrieve entities, attributes, and relationships of an IFC instance file by a user-defined query. Figure 12 represents the user interface of IfcSvr-based validation. This interface allows users to choose necessary concepts and an IFC instance file to selectively execute validation according to the compliance with semantics defined in MVD concepts.

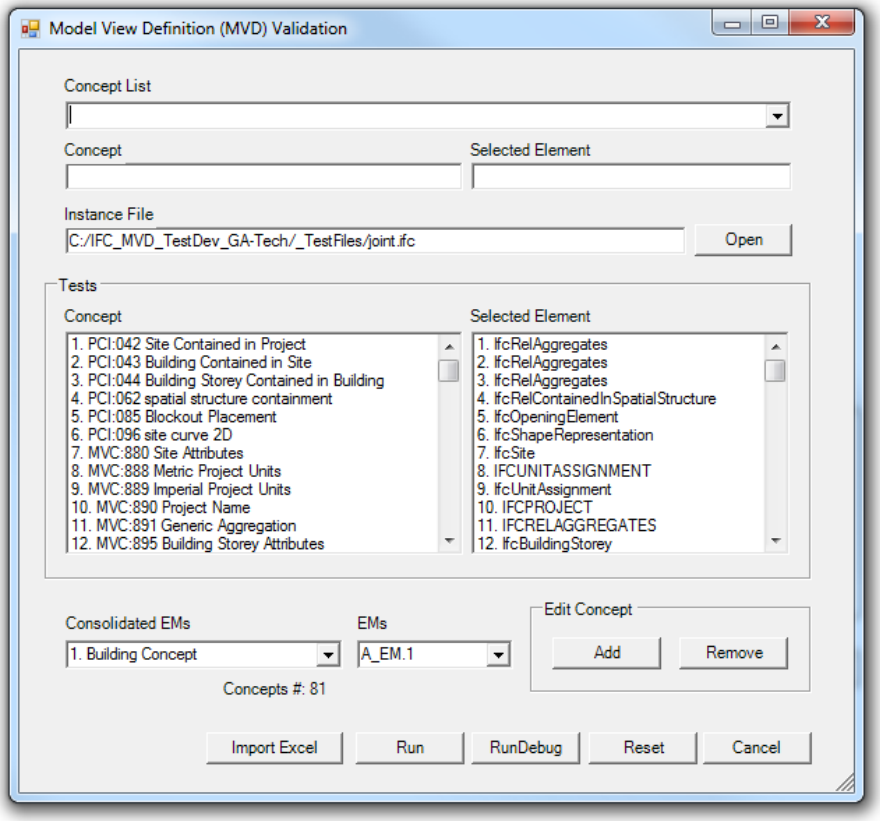


Figure 12 The user interface of IfcSvr-based validation (Lee, Eastman et al. 2015)

Figure 13 shows the validation report generated from the IfcSvr-based checking platform about the requirements of a material association. This platform supports diverse types of rules: the entity inheritance; the accuracy of attribute values and types; the cardinality of attribute values; the types of defined data, and the relationships of corresponding attributes (Lee, Eastman et al. 2015). This tool, however, requires hard-coding for extending checking capabilities and for updating checking modules (Lee, Eastman et al. 2015).

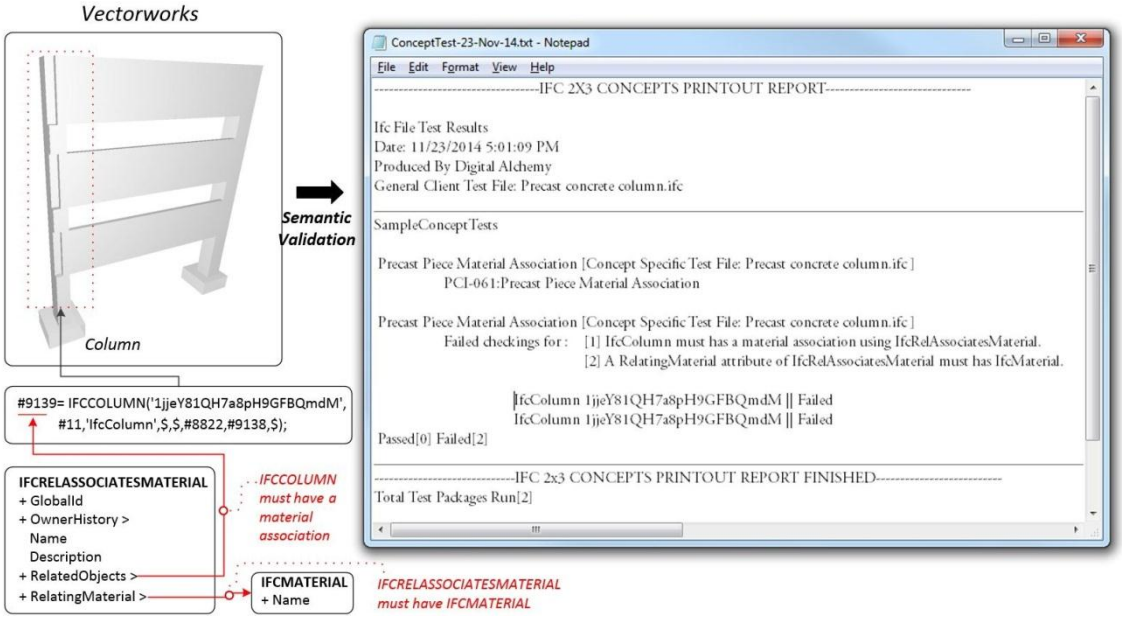


Figure 13 An MVD validation report of IfcSvr-based checking (Lee, Eastman et al. 2015)

The IfcDoc tool, which can automatically create an MVD document, has a validation feature of an IFC instance file according to a corresponding MVD (Gallaher, O'Connor et al. 2004; Gnanarednam and Jayasena 2013). This application, however, provides limited features for evaluating diverse checking scenarios such as a subtype, a referential

relationship, and a semantic content. The mvdXML document contains the mvdXML specifications and corresponding rule sets identified by diverse model views (Chipman, Liebich et al. 2012). Checking scenarios and rules identified from this dissertation research have been provided to the authors of mvdXML document V1.1. In addition, the rule types described in the mvdXML document are matched with some parts of rule types identified from the PCI MVD. Hence, the proposed rule types and implementation scenarios support validation of rule types of the mvdXML document. To ensure its support, the rule logic and checking frameworks would be provided to Model Support Group (MSG) of buildingSMART International. In the active working group, several experts in IFC and an MVD have participated in generating an mvdXML document and developing the IFC specifications. New approach using mvdXML and BIM Collaboration Format (BCF) is one of MVD validation effort. The validation process addresses four types of use-cases identified from the Rdg BIM Norm and Statsbygg BIM Manual (Zhang, Beetz et al. 2015).

Diverse model views have been defined for addressing domain-oriented specifications of a particular industry. Current approaches for MVD checking, however, have limited capabilities to handle diverse types of implementation agreements specified in MVDs. Given an MVD, software vendors and domain professionals must execute evaluation of an IFC instance file according to the specifications of an MVD. In addition, an MVD is the subset of the IFC schema, concepts should be shared by the exchange models of diverse domains. To address these issues, this dissertation has an objective to generalize

rules of model views and to provide a robust methodology of MVD validation with modularized rule logic.

2.4 PRECAST/PRE-STRESSED CONCRETE INDUSTRY (PCI) MVD

From 2008, the PCI BIM Technical Committee and Georgia Tech researchers defined model views for executing functional specifications defined in IDM for a precast concrete domain. The PCI MVD aims to provide the requirements of data exchanges used to translate BIM native objects into an IFC instance file according to the specifications of the PCI industry (Lee, Eastman et al. 2015). Thus, software vendors would use this MVD for developing IFC importer and exporter of their BIM authoring tools. The PCI MVD consists of 96 concept templates reused by several exchange models. Assuming that one concept includes five attributes, the PCI MVD contains about 500 rules that this dissertation analyzed for formalizing rule types.

Figure 14 shows manual testing for precast concrete BIM applications including Tekla, Structure Works, Vectorworks, Scia Engineering, Revit Structure, and Bentley Architecture. To evaluate the accuracy of PCI-bound IFC instance files, software vendors and project participants need an automated validation process that can be iteratively executed to see whether their BIM data fulfill the requirements of data exchanges according to semantics and syntax. A present validation procedure that manually validate IFC interfaces and instance files against an MVD must be improved through an automated and concrete validation process.

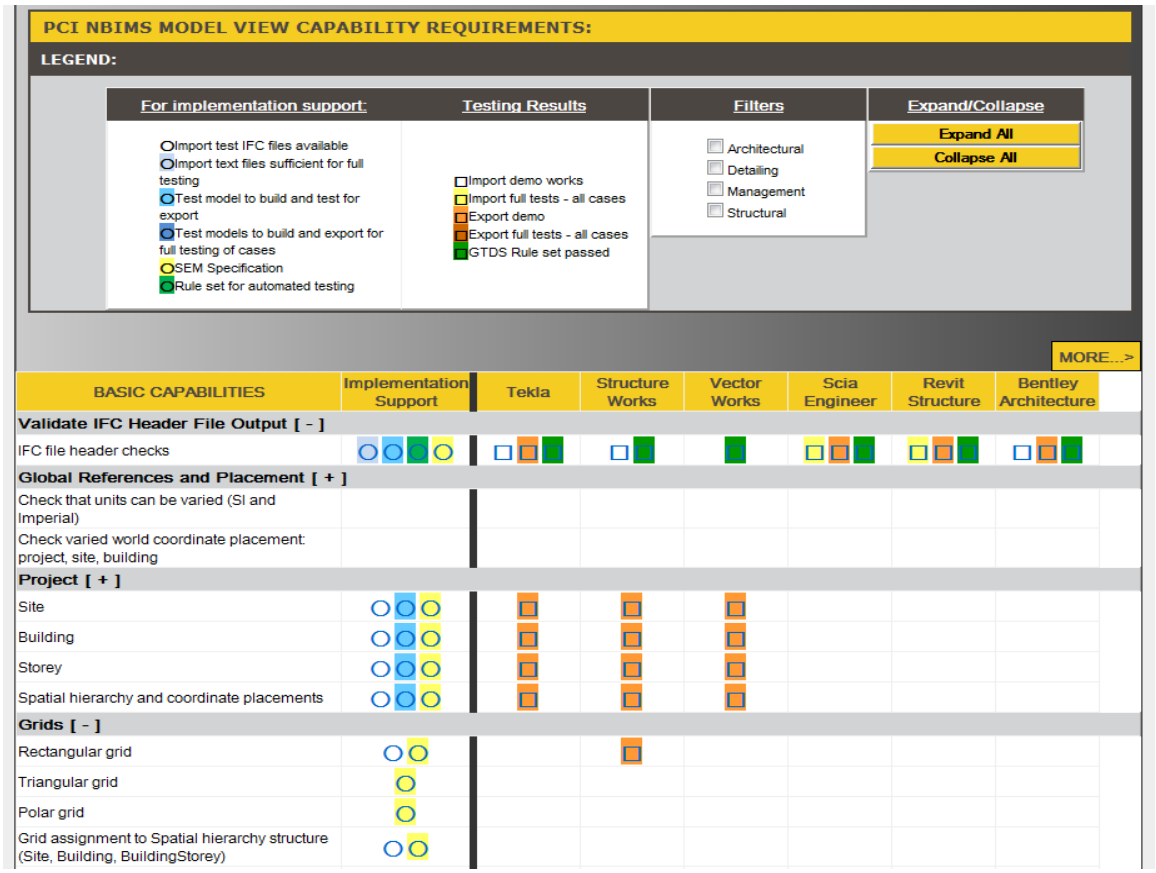


Figure 14 Manual testing for precast concrete BIM applications (Digital Building Laboratory 2012)

CHAPTER 3 MVD RULE LOGIC AND STRUCTURES

Within the defined requirements and restrictions of the IFC schema, an MVD can be defined as its subset to address the domain-oriented specifications of BIM data exchanges. Because an MVD is specified by a set of concepts, they contains relational diagrams and implementation agreements for representing diverse semantics and requirements of a particular BIM data exchange. To provide a robust validation process, validation of an IFC instance file must implement modularized unit testing for each concept. The aggregation of concept tests can execute validation of IFC instance file according to an MVD. To modularize rule types and provide consistent checking frameworks, diverse implementation agreements should be analyzed. IFC instance checking about an MVD primarily includes functional and representational semantics, and syntactic constraints specified in the IFC schema and the EXPRESS language (Lee, Eastman et al. 2015).

To figure out the scope of rule types, this dissertation analyzes the model view of PCI that consists of 96 concepts reused in twelve exchange models (EMs) (Eastman, Sacks et al. 2010). In this dissertation, referring to the PCI MVD, rule types, validation logic, and checking scenarios were extracted and categorized. The rules from such checking components are categorized into diverse types of rule logic and converted to mvdXML templates including corresponding rules. An automated checking procedure implements rule logic on top of the IfcDoc tool. Four general schemes of IfcDoc rule-checking

implementation are representation knowledge, semantic networks, rule logic, production rules and frames.

3.1 Rule types and logic

The objective of this dissertation is to examine types of validation rules categorized from the MVD and to suggest validation logic for each rule set executed on the validation tool. Rule logic is identified from attributes of concept templates modularized to corresponding entities: For example, an aggregation concept template using IfcRelAggregate connecting two IfcElements can be used by subtypes of IfcElement so that they can reuse the predefined aggregation concept template. To formalize rule types of an MVD, diverse rules for attributes defined in a concept were classified as distinct if they require different validation features and checking scenarios. In addition, the author examined concepts and categorized their rules for attributes derived from the 96 concepts of the PCI MVD. Each concept has an average of more than five rules, so the total number of rules reviewed is about 480. The sorted rules were implemented in the validation frameworks, which can address various checking types.

This section describes suggested validation logic and structures that address diverse checking types associated with the specifications of the PCI MVD. Table 1 shows the types of rules identified from the PCI concepts, and Table 2 represents the types of implementation scenarios that apply multiple rule types. Table 1 describes specific types of concept rules and descriptions of each attribute embedded in concepts. Each rule type is expressed by an abbreviation: For example, T refers to accuracy validation of a data

and entity type. Such rule types were defined as rule logic in the first order logic expression. First-order logic is the standardized expression for knowledge formalization broadly used in the mathematics, philosophy, and computer science (Hodges 2001). Using predicates and quantifiers, the first-order logic efficiently express axioms and logical relationships (Avigad et al. 2007). Since such capabilities of the first-order logic have characteristics that are a good match for MVD checking processes, the rule types defined as rule logic in the first-order logic expression are subject to be employed as a backend skeleton for developing validation processes.

Table 2 represents various types of checking executions classified based on checking scenarios. As each attribute demands multiple steps to evaluate diverse requirements, this classification of types of rule implementation shows required rule types for each scenarios. For example, Type-A checking, which evaluates the aggregation data type of an IFC instance file, requires four rule types: U, M, T, and V. SET data type requires that all values in the data type is unique, that the number of its values satisfies the defined number, and that if a required value or type is specified in the attribute, values in the SET data type should fulfill the predefined rules. The concepts include such varied issues as the degree of detail needed, connectivity, aggregation and nesting relationships, type of geometry representation, and other requirements used in distinct workflows (Eastman, Sacks et al. 2010) .

Table 1 Rule types classified by the concepts of the PCI MVD (Lee, Eastman et al. 2015)

Rule Set Type			Type	Note
Checking method	Data value	Check accuracy of a data value	V	Checking a data value
			B	Checking a value with a substring
			A	Checking an arithmetic constraint
			E	Checking an enumeration value
			D	Checking a value of an attribute to be of a defined type (e.g. local placement, coordinate)
		Check cardinality	N	Checking existence or Null of a value
			M	Evaluating lower and upper bound: Setting a limit on the number of attributes
		Check uniqueness of a value	U	Checking global uniqueness within a file
				Checking local uniqueness in aggregation
	Data type		T	Checking the correct type of an entity
			S	Checking subtypes
	References and relationships		R	Checking a referencing relationship
			I	Checking an inverse relationship
	Conditional checking		C	Checking an instance only if a given condition is satisfied
	Syntactic checking		X	Checking the scope of model view and fundamental syntax

Table 2 Implementation types for diverse checking scenarios allowing multiple rule types (Lee, Eastman et al. 2015)

Concept Checking	Note	Type applied
Value-A	Correctness of a simple data value (String: Equal to, contains substring; Integer: equal to, greater, less than)	V, B, A
Value-E	Value-Exist or Null	N
Value-U	Value-Uniqueness of a value (Global, Local)	U
Value-S	Value-Selecting one of enumeration values	T, V
Value-C	Type-Conditional value checking of an attribute	R, I, V
Type-C	Correctness of a simple data type (Referenced, Referencing data type)	T
Type-A	Type-Aggregation data type (Set, List, and Array)	U, M, T, V
Type-E	Type- Selecting one of enumeration values (Select data type)	S, T
Type-S	Type-Subtype	S
Type-D	Type-A value of defined data type	D
Type-O	Type-Optional type (1) Null (2) Correct value (Single, Aggregation, Enumeration, Subtype)	N, V, T
Type-R	Type-Conditional type checking of an attribute dependent on references	R, V, T
Type-I	Type-Inversely related	I

3.2 Rule structure and execution methods

According to the domain of interest, users define model views using data models defined in the EXPRESS language that help to formally generate data objects and relationships. The EXPRESS language, which is formalized in the ISO Standard for the Exchange of Product model STEP (ISO 10303) and standardized as ISO 10303-11 is a

standard data modeling language. The EXPRESS language is the foundation of terminology and expressions used to represent rule logic and structures.

This section provides diverse types of rule sets and concept examples of the PCI MVD with regard to rule types in Table 1: (1) data accuracy, (2) sub-string checking, (3) arithmetic constraints, (4) enumeration values, (5) entity data type, (6) subtype checking, (7) reference/inverse relationships, (8) existence and null, (9) cardinality, (10) uniqueness, (11) conditional checking, and (12) fundamental syntactic checking. In addition, this chapter provides an introduction to description logic as a formal language for representing knowledge and reasoning about.

3.2.1 Data accuracy

This checking type is associated with the semantics of a building information model required by domain professionals for data exchange. Model views enable domain experts to define requisite data for the attributes of entities such as a name, a description, an object type, a representation type, and a tag (Lee and Eastman 2015). Such attributes are also defined in corresponding data format such as IfcLabel, IfcText, or IfcIdentifier for a specific purpose. These data types are STRING, BINARY, BOOLEAN, and LOGICAL. Based on the name of attributes that declare their roles, they can include diverse kinds of data: explicit, derived, and inverse attributes. Explicit attributes are values directly embedded in an instance and shown in a P21 file, derived attributes are delivered by the schema such as using an expression, and inverse attributes represent values by being referred from other entities. The values of these attributes are articulated

in IfcLabel or IfcText entities. Because most attributes represented by IfcLabel or IfcText entities are optional for a P21 instance file, domain professionals are able to define corresponding values for such attributes to define data exchange requirements. Thus, the values of these attributes that are determined based on specific purposes can be used as criteria to validate an IFC instance file pertaining to accuracy. Accuracy checking is the pivotal and explicit rule type required for implementing diverse rule structures. The logical expression shows a semantic comparison of an attribute value with one defined in model views.

(1) Logical expression (Lee and Eastman 2015)

- $\forall x \forall y(\text{Entity}(x) \wedge \text{ValueAttribute}(x, y)) \equiv \text{ValueMVD}(x, y)$
- $\forall x \forall y(\text{Entity}(x) \wedge \text{ValueAttribute}(x, y)) \neq \text{ValueMVD}(x, y)$

This logic expression borrows the description logic description of first rule logic. x is an entity and y is its attribute. $\text{ValueAttribute}(x, y)$ retrieves a value of a y attribute of a x entity. $\text{ValueMVD}(x, y)$ finds a value of a y attribute of a x entity from a model view definition. Data accuracy checking is also used to determine whether or not the following step of the execution of dependent rules proceeds. For example the (2) Logical expression below, if an operand is true, $\text{SubordinateChecking}(x, y)$ implements subordinate rule-sets defined underneath a corresponding entity. Thus, in case of a true outcome, a valueOf() method dependent on a root entity is called and executed for comparing the following instances.

(2) Logical expression (Lee and Eastman 2015)

- $\exists y \forall x((\text{Entity}(x) \wedge \text{ValueAttribute}(x, y) \equiv \text{ValueMVD}(x, y)) \rightarrow \text{SubordinateChecking}(x, y))$

3.2.2 Sub-string checking

A model view encompasses diverse types of values defined by domain professionals. The American Institute of Steel Construction (AISC) defines more than several thousand types of a reinforcing bar defined by diverse organizations and nations. The PCI also defines various types of a wall that a P21 instance must fulfill one at least. Such an instance file, however, might not satisfy the listed types because the heterogeneous IFC translation processes of BIM authoring tools can provide various names for the wall. In addition, the names of such values could be distinctly used by various domains. For example, a wall cladding can be called by a covering, a barrier, a wall cover, and others. Thus, adding a possible list of values is unlimited.

To resolve this issue, government representatives, public organizations, and industries have developed standards and dictionary libraries such as the International Framework for Dictionaries Library (IFD Library) or the OmniClass (Leite and Akinci 2011). In particular, the IFD library developed by the group of the International Alliance for Interoperability (IAI) improves flexibility for data exchange of IFC files linked with model databases. The dictionary standard is imperative to support an integrated technology and to benefit exchange of building models, providing generalized object-

oriented data set across industry domains. Even though such standards offer model enrichment that allows for advanced dictionaries, because they have insufficient capabilities that cannot address newly emerged building information, this substring checking would be useful to identify required data from a P21 file. In the logical expression below, the $\text{SubStringValueMVD}(x, y)$ function identifies a value of a y attribute of a x entity from a model view definition and lists the defined substrings according to the targeted attribute.

(3) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big((\text{Entity}(x) \wedge \text{ValueAttribute}(x, y) \in \text{SubStringValueMVD}(x, y)) \big)$

3.2.3 Arithmetic constraints

Arithmetic constraints contain REAL, INTEGER, and NUMBER of simple data types. In accordance with the demands of diverse domain experts, data for this type of checking involve a value, a size, and a type. Such data types are defined and evaluated by the setting of an operator that includes Equal, Not Equal, Greater Than, Greater Than or Equal, Less Than, Less Than or Equal, and Included in. Using such operators for a numerical comparison, we can analyze the attributes of the entities in an IFC instance file. Equal and Not Equal operators help users evaluate the accuracy of corresponding data as shown in the logical expression of a simple comparison. Greater Than, Greater than or Equal, Less Than, Less Than or Equal operators enable users to compare defined and instance values by signs of inequality.

(4) Logical expression (Lee and Eastman 2015)

- $\forall a \forall x \big((\text{Entity}(y) \wedge \text{ValueAttribute}(x, a) > \text{ValueMVD}(x, a) \big)$
- $\forall a \forall x \big((\text{Entity}(x) \wedge \text{ValueAttribute}(x, a) \geq \text{ValueMVD}(x, a) \big)$
- $\forall a \forall x \big((\text{Entity}(x) \wedge \text{ValueAttribute}(x, a) < \text{ValueMVD}(x, a) \big)$
- $\forall a \forall x \big((\text{Entity}(x) \wedge \text{ValueAttribute}(x, a) \leq \text{ValueMVD}(x, a) \big)$

3.2.4 Enumeration values

The IFC specifications define diverse enumeration values for an attribute that are simple strings or types. For instance, IfcObject can entail a type using the inherited PredefinedType attribute, which is a predefined label in the IFC schema. Such attributes denote particular types that can be defined at the level of instantiable subtypes. The comprehensive list of predefined types for IfcWall consists of eight options defined in the IFC schema. Thus, to represent a type of a wall, a P21 instance file can choose one of the enumeration values.

In addition, the Included, one of operators, encompasses several values so that users add acceptable data for an attribute and evaluate if an instance file has one of the listed values. In addition, the IFC schema defines enumeration values for diverse attributes. $EnumerationValueSchema(x, y)$ retrieves a list of enumeration values defined in the schema. Within such defined enumeration values, users can add one layer to restrict the possible options pertaining to their own requirements of data exchange. $EnumerationValueMVD(x, y)$ helps add user-defined values into an enumeration value list subject to be compared with instance values.

(5) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(Entity(x) \wedge ValueAttribute(x, y) \in EnumerationValueMVD(x, y) \wedge EnumerationValueSchema(x, y) \big)$

3.2.5 Entity data type

The IFC specifications define specific entity data types for specific attributes so that an IFC instance allows only predefined types. With this baseline, users can narrow down the acceptable entity data types for an attribute by defining model views. Thus, evaluating defined entity data types is a fundamental and important procedure that ensures the accuracy of data models. Such a checking type is closely related to supertype and subtype checking according to attributes.

(6) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(Entity(x) \wedge \text{ReferenceAttribute}(x, y) \equiv \text{EntityType}MVD(x, y) \big)$

3.2.6 Subtype checking

The design of the IFC schema assumes a several-layered hierarchy, allowing multiple inheritance. Thus, we can define an entity as a supertype or a subtype, and one supertype entity can include several subtype ones. Diverse subtype entities of a defined root entity can comprise an attribute. For example, if an aggregation concept states the RelatingObject attribute of IfcRelAggregate as IfcWall and RelatedObjects as

IfcBuildingElement, the subtype entities of IfcBuildingElement such as IfcColumn can satisfy the RelatedObjects attributes. Hence, subtype checking must be integrated into the reference validation rule.

(7) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(Entity(x) \wedge \text{ReferenceAttribute}(x, y) \subset \text{EntityType}MVD(x, y) \wedge \text{SubEntityTypes}(y) \big)$

3.2.7 Reference/Inverse relationships

An IFC instance file has a complex structure that consists of diverse references including inverse relationships. Such available relational structures have been already defined by the IFC schema. Within such a limitation, an entity relationship can be determined by the specifications of data exchange. Thus, an attribute must refer to correct entities and have acceptable inverse relationships.

(8) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(Entity(x) \wedge \text{ReferencingAttribute}(x, y) \equiv \text{ReferenceEntity}MVD(x, y) \big)$
- $\forall y \forall x \big(Entity(x) \wedge \text{ReferencedAttribute}(x, y) \equiv \text{InverseReferenceEntityMVD}(x, y) \big)$

3.2.8 Existence and Null

One critical factor for exchanging data is validation of the existence of requirements. Even though specific values or types are not defined for an attribute, domain professionals are able to require the existence of a corresponding value. For example, all objects must have the 22 length string for a GUID attribute, and the structural engineering requires the values of a structural load, a stiffness, a pressure, and a wind analysis. Such existence checking can be applied to three levels: an attribute, an entity, and a relationship. For an attribute, existence can be defined using cardinality: The lower bound is one. Because the IFC specifications set mandatory and optional rules for an attribute, users can define this condition in accordance with their purposes. Existence checking for an attribute is validation that evaluate whether a specific attribute satisfies mandatory and optional requirements.

(9) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x (Entity(x) \wedge \text{NumberOfValueAttribute}(x, y) \geq 1)$
- $\forall y \forall x (Entity(x) \wedge \text{ValueAttribute}(x, y) \equiv \text{Null})$

For an entity, adopting existence checking might result in a problem. The ISO 10303-21 defines several required IFC entities that a part 21 instance file must include such as an IfcProject or IfcOwnerHistory representing metadata. The main issue, however, lies in that existence for an entity specifies that a building information model must have a certain type of an object. For example, if a model view defines IfcWall mandatory for data exchange between an architect and a constructor, is a building model that lacks

IfcWall an inaccurately designed one? How can domain professionals define that a building model used in certain data exchange must encompass a specific object instance? Because the IFC specifications represent building data used for data exchange not for modeling and object representation, such schema allow diverse interpretations and representation types. Thus, a wall can be represented as IfcWall, aggregated IfcBeam, or IfcElementAssembly. Such elements act as a wall but use various representation methods. Thus, defining the existence of a certain object type and reporting an error pertaining to its omission must be carefully addressed.

For a relationship, existence can be defined to illustrate a desirable relational structure required by specific data exchange and modeling application used by associated domain professionals. This condition is imperative to satisfy minimum requirements of IFC translation because diverse software companies have heterogeneous IFC interfaces used as import and export features. For example, a representation type such as a boundary representation type can be defined in a variety of approaches in the IFC specifications. To analyze and translate into or from the IFC instance files, software companies need to cover all distinct relational structure that can be defined in the IFC specifications. Thus, defining requirements for a necessary relational structure would be helpful for software vendors to understand and transform IFC instance files. Such a rule for a relationship, however, is accompanied by conditional checking for a root entity: If a certain entity exists, a corresponding relationship must be generated in a P21 file.

(10) Logical expression (Lee and Eastman 2015)

- $\exists y \forall x \big((\text{NumberOfEntity}(x) \geq 1) \rightarrow (\text{ReferenceAttribute}(x, y) \geq 1) \big)$

With regard to defining values for an attribute of an object instance, some attributes must be Null because they are not used at all based on domain knowledge. In addition, because some attributes are unnecessary to represent native objects of BIM authoring tools, they can be skipped to shrink file size. Thus, Null checking, which is to identify if a corresponding entity has a value for its attribute or not must be employed to validate instance files pertaining to requirements of a model view. The attribute, however, can encompass a Null rule within the range of the IFC specifications. In other words, this Null checking is only defined to an attribute that has zero as the lower bound in the IFC specifications.

(11) Logical expression (Lee and Eastman 2015)

- $\exists y \forall x \big((\text{NumberOfEntity}(x) \geq 1) \vee (\text{ReferenceAttribute}(x, y) \geq 1) \rightarrow \text{NumberofCorrespondingReferenceAttribute}(x, y) \equiv 0 \big)$

3.2.9 Cardinality

Cardinality, which defines lower and upper bounds of an attribute and a relationship, is an imperative checking type. The IFC schema specifies cardinality for all attributes as a syntactic restriction so that building model data are efficiently represented

and managed. Because a model view is a subset of the IFC schema, user-defined values must follow this lower and upper bound rules. Within the available range of cardinality defined in the IFC schema, users can decide appropriate cardinality for an attribute based on an exchange requirement. For example, IfcRepresentation has Items, which can refer IfcRepresentationItem with a cardinality, SET [1:?] defined by the IFC 2x3 schema. Even though the lower bound is one, the upper bound that this attribute can refer is unlimited. If data exchange defines three as the upper bound, SET [1:3] is applied as a requirement for this attribute. Thus, this condition satisfies both the IFC schema and a data exchange requirement.

Two types, N and M, are underneath cardinality checking. One different thing is that the N type is checking of existence and null of a value. For example, (1) ObjectType of IfcWall must exist: checking whether a value exists as IfcLabel or not. (2) BarRole of IfcReinforcingBar must be null: checking whether a value is as IfcLabel null or not. The M type is checking of arity of relationships. For example, Coordinates of IfcCartesianPoint need one relationship of IfcLengthMeasure as lower bound and its three relationships as upper bound. Thus, this type calculates the number of relationships.

(12) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(\text{NumberOfLowerBoundReference}(x, y) \leq \text{NumberOfReference}(x, y) \leq \text{NumberOfUpperBoundReference}(x, y)\big)$

In particular, all of the inverse relationship has [0:?] cardinality setting. In case of that the lower bound is zero, this attribute is optional. In other words, if the lower bound is greater than one, this attribute is mandatory. Manipulating the required number of an attribute can manage the optional and mandatory setting, which is critical to validate instances.

(13) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big(\text{NumberOfLowerBoundReference}(x, y) \geq 0 \rightarrow \text{OptionalChecking}(x, y)\big)$
- $\forall y \forall x \big(\text{NumberOfLowerBoundReference}(x, y) \geq 1 \rightarrow \text{MandatoryChecking}(x, y)\big)$

3.2.10 Uniqueness

As defined in the IFC schema, all object instances typically require a globally unique identifier (GUID) attribute, which follows the universal unique identifier standard (UUID). The GUID is compressed and employed for the objective of data exchange according to a published compression function of GUID generation algorithms in order to reduce overhead required by all IFC object instances. Thus, from IFC R1.5.1, the identifier has been shrunk to 22 characters through an algorithm, which binds the bits of an identifier onto a base 85 characters as follows:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn

opqrstuvwxyz!#$%&^|+,-./:;<=>?~`@_*

As defined in the IFC schema, TYPE IfcGloballyUniqueId = STRING (22) FIXED, now a fixed 22 character length string must be maintained by the P21 file. This GUID is significantly useful to conserve a space while physically exchanging building information models among diverse BIM applications. To satisfy this uniqueness rule, object instances in a P21 file must include 22 length unique identifiers for a GUID attribute. Thus, each GUID can be evaluated for its uniqueness through comparing all of GUID of object instances encompassed in a P21 file. This uniqueness rule set can be assigned to any attributes that can include a value in IfcLabel or IfcText data types. For example, a model view can define the uniqueness rule for the TAG attribute of one specific building element. Similar to the validation about GUID, this checking process executes a comparison between a corresponding value and all relevant values defined in the Tag attributes.

(14) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big((\text{StringLengthOfEntityGUID}(x, y) \equiv 22) \wedge \text{UniqueValueInAllObjects}(x, y) \big)$

The uniqueness checking is required at a local level, where a collection exists as SET data type. SET is defined in a cardinality setting to store unique objects and values within one attribute. Thus, an attribute defined as SET is not allowed to contain duplicate elements. Uniqueness for a SET data type can be evaluated using a collection of values contained in one attribute.

(15) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big((\text{ArityEntity}(x, y) \equiv \text{SET}) \wedge \text{UniqueValueInEnumeration}(x, y) \big)$

IfcGloballyUniqueId representing the GUID and uniqueness of SET and LIST cardinality can be validated in the level of syntax. These requirements, however, are invaluable in exchanging data. Despite an obscure scope of validation between syntax and semantics, this dissertation considers some syntactic requirements as model view specifications as well. Such an issue pertaining to the proper scope of model view validation will be discussed in Section 7.

3.2.11 Conditional checking

Validation implementation frequently consists of diverse types of rule sets. Some rules are quite dependent on other ones. Thus, based on validation outcome of each rule, subordinate rules must be controlled and managed pertaining to their executions. For example, where if a predefined condition is not satisfied for the relational structure, then the constraint check would not run and a failure would be reported. In other words, if the validation returns false, the dependent rules stop for corresponding entity.

(16) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x \big((\text{CheckingResult}(x, y) \equiv \text{True}) \rightarrow \text{SurbodianateChecking}(x, y) \big)$
- $\forall y \forall x \big((\text{CheckingResult}(x, y) \equiv \text{False}) \rightarrow \text{StopCorrespondingCheckingProcess}(x, y) \big)$

The condition can be defined using parameters that an instance can have. For example, users can define several parameters for an ObjectType attribute of the IfcWall such as precast concrete. Based on predefined parameters, dependent rule sets should be defined such as RepresentationType including SweptSolid or Boundary Representation.

(17) Logical expression (Lee and Eastman 2015)

- $\exists y \forall x \big(((\text{ValueAttributeofEntity}(x, y) \equiv \text{True}) \in \text{ParametersOfAttribute}(x, y)) \rightarrow \text{SurbodianateChecking}(x, y) \big)$

For this rule, one particular thing needed is the order of rules. Conditional parameters should be evaluated before any other constraint parameters. Regardless of any defined ordering, condition rules are always checked before constraints.

3.2.12 Fundamental syntactic checking

A model view is a subset of the IFC schema. Thus, a P21 instance file typically follows specifications defined the IFC schema and the EXPRESS language. Diverse tools such as the Express Engine and the EXPRESS Data Manager™ evaluate a P21 file in terms of accuracy of syntax. When evaluating a P21 instance file, we assume that a P21 file is syntactically correct because current validation process is too fragmented to ensure integrity and interoperability of an IFC instance file. Ultimately, an integrated approach, which supports syntax and semantic validation, is preferable and desirable. To address this issue, fundamental syntactic rules should also be included as a required checking element. The checking logic may invoke EXPRESS functions and procedures in order to formulate complex rule statements with instances, constructs, parameters, and constants.

Rule checking works based on compiling the schema and loading instances – if an instance does not conform to the schema, then downstream rule checks may report erroneous results. I presume the intent here is to include the entire IFC schema within scope such that if a file contains an instance of an entity or attribute that has not been referenced within the model view, it is still considered within scope such that no schema checking errors occur. IfcDoc 8.9 adds an option for any Model View, where there's a checkbox indicating Include All Definitions – with this option checked, then the entire IFC schema is within scope.

(18) Logical expression (Lee and Eastman 2015)

- $\forall y \forall x((\text{ModelViewDefinitions} \subset \text{IfcSchema}) \wedge (\text{RulesOfConcepts} \subset \text{IfcSchema}) \wedge (\text{RulesOfConcepts} \subset \text{EXPRESSLanguage}))$

Constructed data types can be defined within an EXPRESS schema. They are mainly used to define entities, and to declare type of entity attributes and aggregate members. Data types can be used in a recursive way to build up more and more complex data types. For example, it is possible to define a LIST of an ARRAY of a SELECT of either some entities or other data types. If it makes sense to define such data types is a different question. EXPRESS defines a couple of rules how a data type can be further specialized. This is important for re-declared attributes of entities.

In summary of this sections, since the proposed rule logic of model views is a list of fundamental principles and rule descriptions, they are subject to be used iteratively in developing rule features and implementing rule checking processes. To execute rule checking of one attribute, several types of rule logic should be composed for addressing various types of P21 instance files that entail complex IFC hierarchy and flexible representation types. In this dissertation, the list of rule logic was employed to execute rule checking processes of (1) modularized IfcSvr-based MVD validation and (2) the IfcDoc tool.

CHAPTER 4 IMPLEMENTATION OF RULES

4.1 IfcDoc application

IfcDoc is an open source Building Smart International software application supporting various functions such as creating and documenting the MVD for software developers and data integrators working with IFC. The primary goal of IfcDoc is to help users generate an IFC MVD document that includes the diagrams, schema definitions, and contents generated based on the user-defined configuration (buildingSMART 2015). Rule checking of model views is an extension for IfcDoc to support automated testing that assesses if a candidate IFC instance file is correct in relation to a target MVD. The tool was updated so that it contains the rule logic and structures found in the PCI MVD.

Using IfcDoc, software vendors and domain experts can evaluate an IFC instance file according to the following concept descriptions: (1) adding concept templates in the IfcDoc, (2) drawing a relational structure and generating rules, (3) assigning them to entities and choosing mandatory and optional settings for each exchange, (4) executing a validation feature with a pertinent exchange model, and (5) identifying errors illustrated in the HTLM and user interface reports and debugging an IFC instance file. The MVD can be mapped into mvdXML format that can provide and support MVD validation with defined rules (Lee, Eastman et al. 2015). Figure 15 shows the UI of IfcDoc. The template page supports defining rules for attributes and supported entities.

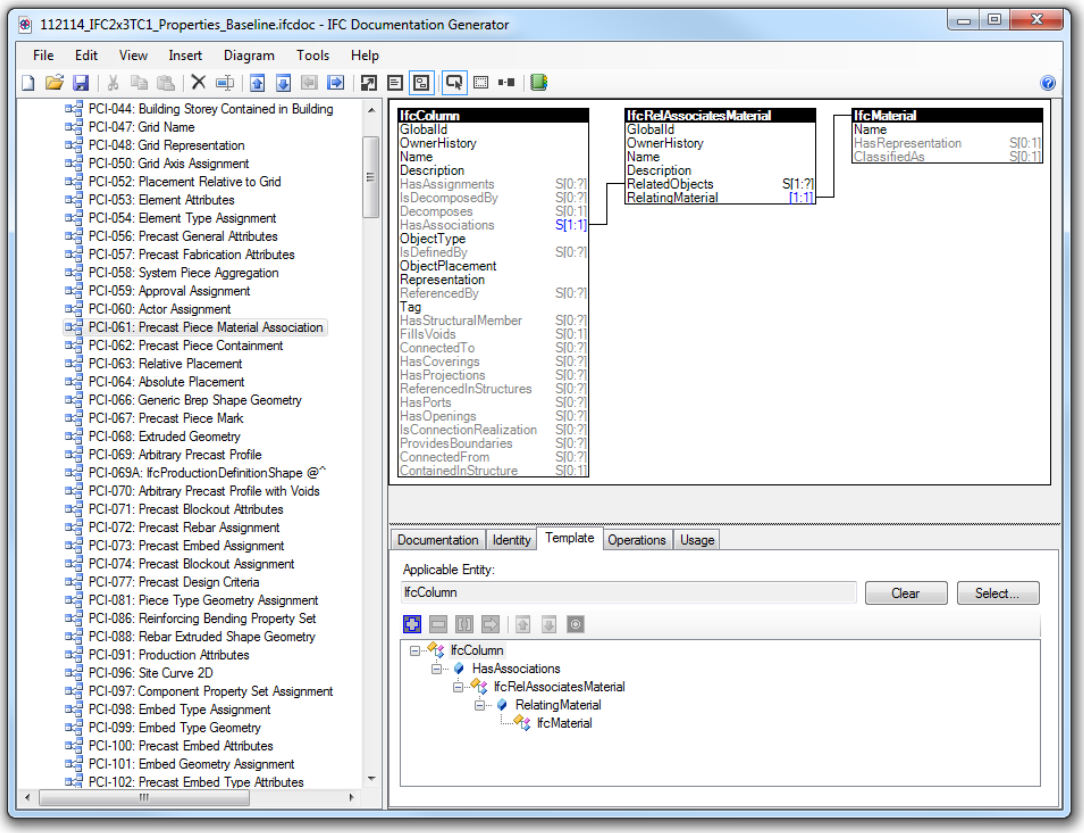


Figure 15 The user interface of the IfcDoc tool (Lee, Eastman et al. 2015)

In addition, Figure 16 represents validation reports in the HTML format. This report shows the error of a material association pertaining to a precast concrete column model exported from Vectorworks. Since the sample file does not refer to IfcMaterial as a RelatingMaterial attribute of IfcRelAssociateMaterial to represent a material relationship, this report generated a fail result regarding such validation.

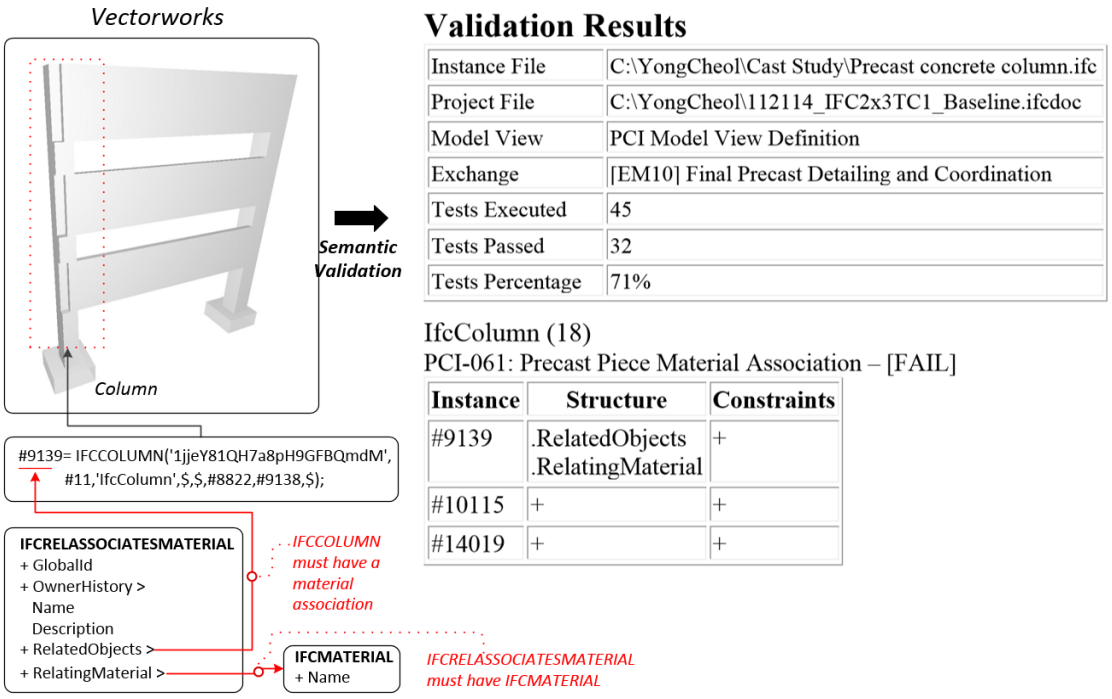
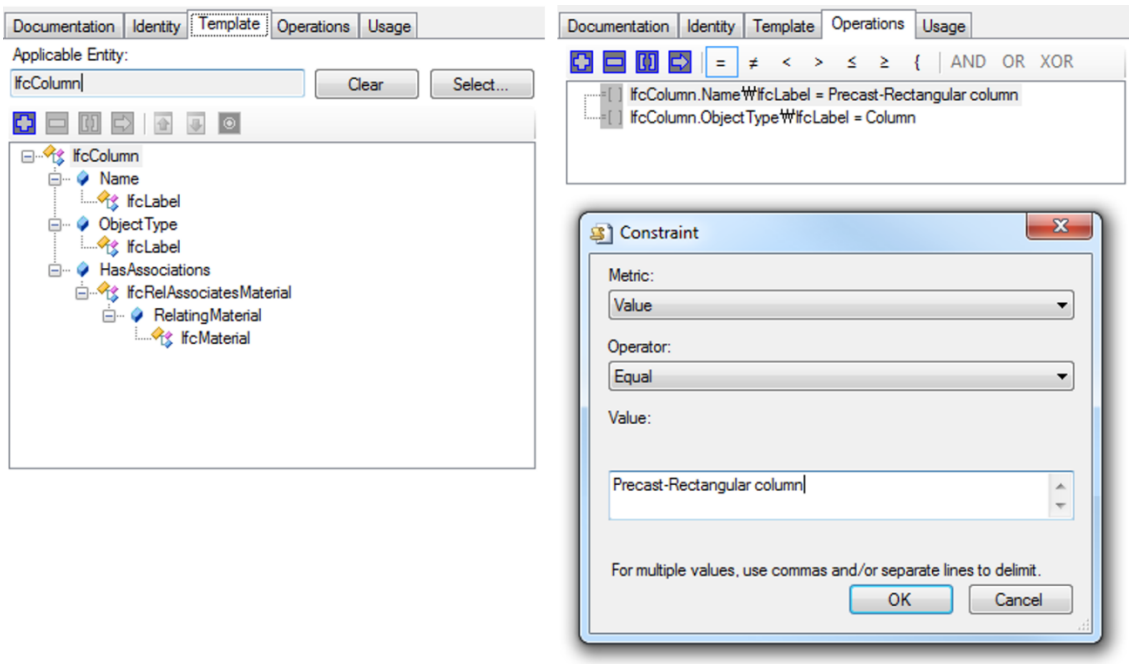


Figure 16 An HTML validation report of the IfcDoc pertaining to the PCI-061 precast concrete concept (Lee, Eastman et al. 2015)

The IfcDoc tool offers a method that allows users to define concept documents. To promote reusability of concepts, a form of a template can be generated and then assigned to entities that need to use it. Typically, a concept template is defined with a root entity that can be a higher-level entity of anticipated entities because the availability of this concept assignment is determined based on the root entity. For instance, a concept template that has IfcBuildingElement as a root entity can be used by its subtype entities, but cannot be employed by entities that exist in a different category not associated with IfcBuildingElement such as IfcDiscreteAccessory. Thus, considering an appropriate root entity is an important factor in defining concept templates on the IfcDoc tool. In addition, at a template level, the requirements of a root entity including an explicit attribute, which

involves direct values visible in a STEP file, can be re-declared at a dependent template level. In terms of derived attributes, they obtain values from an expression, which mostly refers to other attributes through EXPRESS functions. Inverse attributes can only name and constrain an explicit attribute to a corresponding entity without having any added information to an entity.

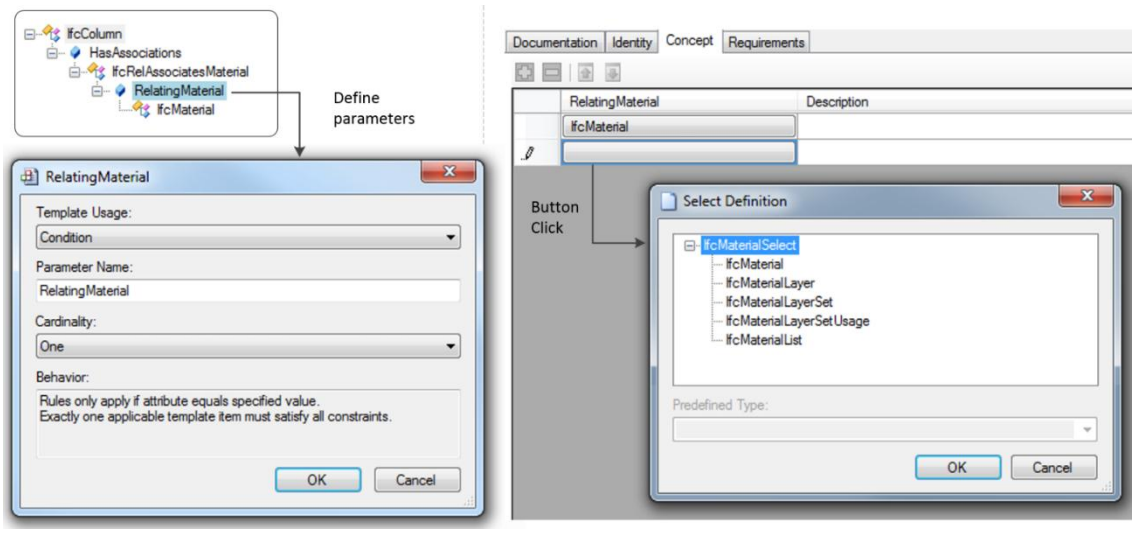
The IfcDoc interface allows users to define diverse types of rules. In terms of a structure including relationships and references, when users generate a diagram on IfcDoc, its relational references are defined as criteria for validation. Figure 17 shows the Template category that enable users to specify the relational requirements of IfcColumn. To generate constraint rules for specific values and properties, users should select pertinent values for Metric and Operator and input semantics for associated attributes. Figure 17 shows the defined constraint that a Name attribute must be a "Precast-Rectangular column" value. In addition, for an ObjectType attribute, the IfcColumn entity of IFC instance files must include a "Column" value (Lee, Eastman et al. 2015). For types of metric parameters, Value, Size, Type, and Unique, are provided. In addition, users can use diverse the operator parameters: Equal, Not Equal, Greater than, Less than or Equal, and Included (Lee, Eastman et al. 2015).



(a) Rule for a structure (b) Rule for a constraint and its user interface

Figure 17 An interface for defining the rules of a structure and a constraint (Lee, Eastman et al. 2015)

IfcDoc provides a parameter feature that allows users to define a parameter for one attribute or relation. As shown in Figure 18, when a user define a parameter for RelatingMaterial as a Condition setting, IfcDoc enables users to add various options to the attribute. In other words, this feature helps users apply multiple entities and values into one attribute checking. This interface has features for a template use, a parameter name, cardinality, and behavior (Lee, Eastman et al. 2015).



(a) Interface for defining column headers of rules (b) Interface for defining conditions of values of rules

Figure 18 The definition process of a rule parameter in the IfcDoc (Lee, Eastman et al. 2015)

In summary, the IfcDoc tool, which was originally designed to provide an MVD document, has been developed to address diverse rule types identified from this dissertation. Validation of IFC instance files and interfaces concerning the requirements of concept descriptions is the second current use of IfcDoc. The light-weight UI and rule checking features of IfcDoc are expected to support to efficiently define and manage relevant concept documents and their constraints. The most important benefit of use of IfcDoc as a checking platform is that IfcDoc allows users to not only define an MVD document, but also employ it as reusable and electronically shareable checking criteria.

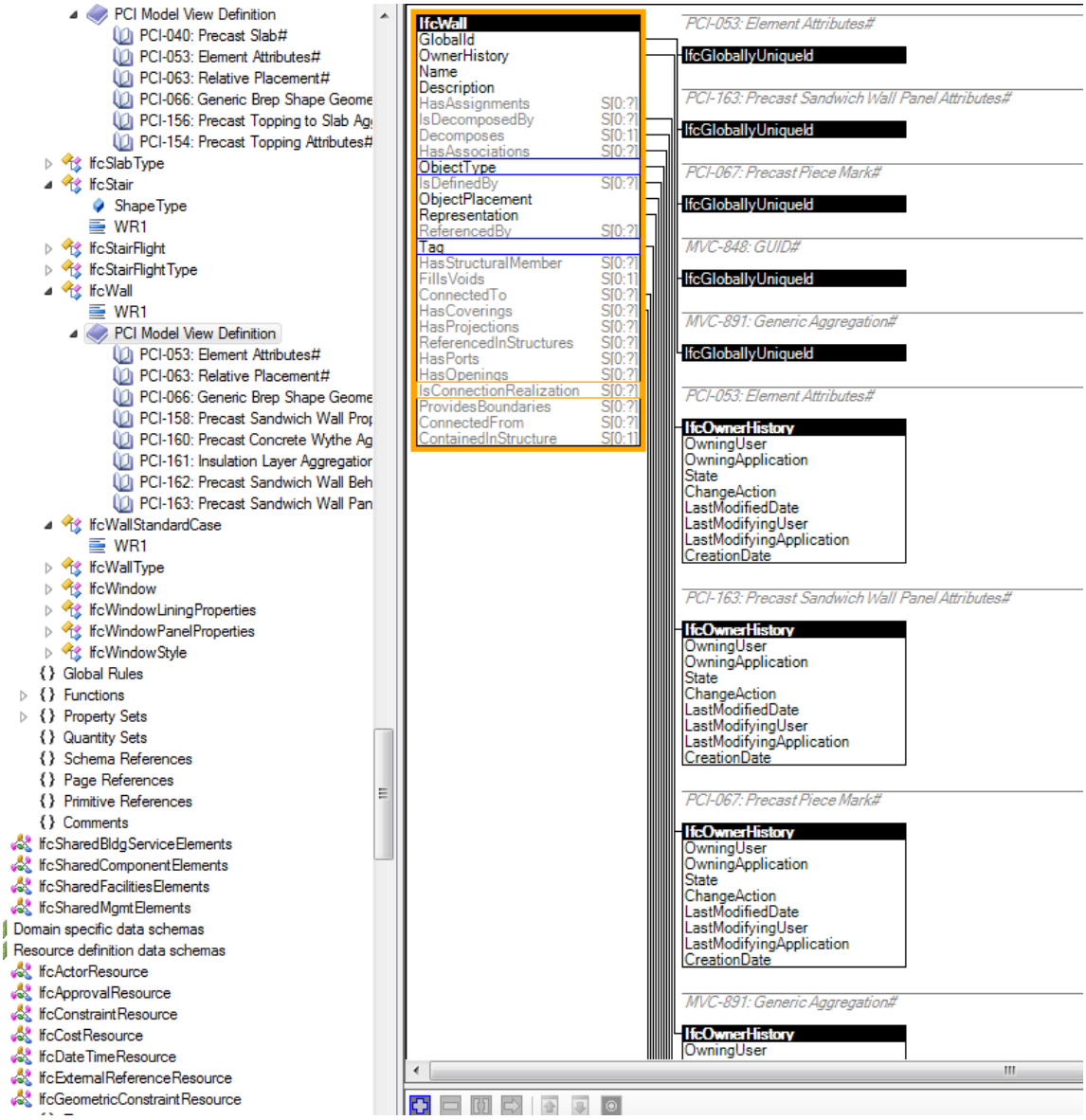


Figure 19 All concepts defined under IfcWall

4.2 Rule structure and implementation methods

The effort to implement model views so that exchanges can be solid and robust is a continuing issue. Rule logic identified from the PCI MVD were coded for implementation on IfcDoc. The developed checking features of the IfcDoc tool can handle various execution scenarios of MVD validation. Figure 20 shows the research processes of this dissertation regarding an efficient methodology with modularized rule logic and a robust platform for MVD validation. To identify the types of rules, this dissertation used the PCI MVD. This architecture represents processes and methods of generating rule logic and a validation framework and implementing them using the IfcDoc tool. The rules of each attribute in concepts are categorized as rule logic. Such rule logic plays a pivotal role as a framework for defining and executing rules on the IfcDoc application. Furthermore, while validating instance files, rule logic are implemented by a MVD validator of the IfcDoc platform. One other benefit of the rule modules is to allow users to reuse existing concept description. Even though, concepts for MVDs are designed for improving reusability, it was difficult to implement it because of lack of concept retrieval system that helps users to find appropriate concepts. However, rule logic can be an identifiable key that help to provide similar concepts with proposed concepts based on the rule types and definitions. In addition, rule logic can be used in a recursive way to build up more complex rule sets implemented in the validation application.

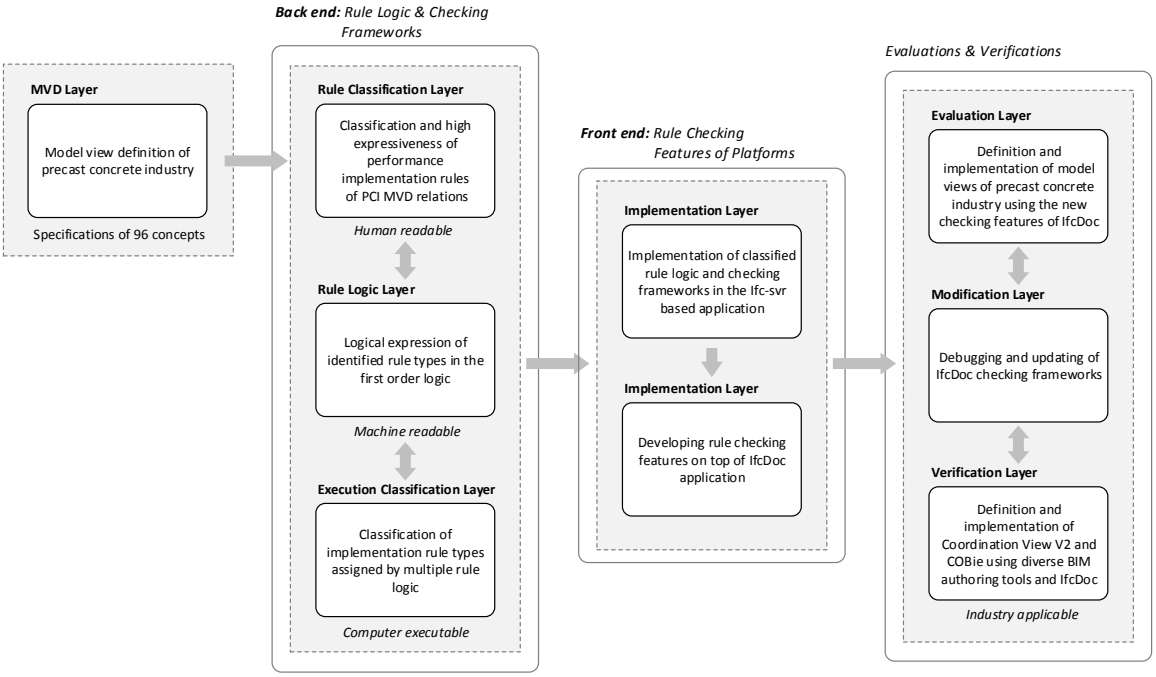


Figure 20 Processes of research for developing modularized rule logic and a robust platform for MVD validation

This section describes types of rule-checking implementation and corresponding validation process and rule types. Each rule implementation utilizes thirteen rule types illustrated in Table 1. In addition, for explaining several cases identified in this dissertation, examples retrieved from the PCI MVD are provided with regard to ten implementation categories in Table 2: (1) data accuracy, (2) cardinality, (3) uniqueness, (4) reference/inverse relationship, (5) relations based on types of objects, (6) conditional checking, (7) geometry representation types, (8) property sets, (9) mandatory and optional setting, and (10) validation throughout requirements of multiple concepts. Each section shows general rules of the PCI MVD and corresponding rule logic used for executing rules. The examples represent what types of rule logic are applied to implement

general rules. In addition, to show implementation of diverse rules, each table shows defined rules in the IfcDoc interface using rule logic. This section contains detailed descriptions for ten rule types and associated rule logic to demonstrate the intentions of general rules and their implementations.

4.2.1 Data accuracy

The MVD governs a data exchange process to guarantee interoperability of building data and to facilitate collaboration of industries. For well-organized requirements of the MVD, domain professionals declare a variety of detailed specifications including values and data types defined for a particular entity and attribute. The primary objective of MVD validation is to evaluate such semantics of a P21 file according to diverse semantic requirements for data exchanges. In addition, checking implementation consists of combined checking types including data accuracy checking at least once. Thus, data accuracy checking is a fundamental element for most of rule implementation. In addition, since most attribute checking requires multiple rule types, correctness checking type is frequently used in comparing instance values with defined ones of a concept. For example, Table 3 represents accuracy checking of a simple data value and type.

Table 3 The correctness of a simple data value and type

<table>
<tr><td>Requirements</td><td>The concept PCI-053, which defines building element attributes, is used by several precast or non-precast building elements. The value for ObjectType, which represents the generalization of any semantical entity type and must be an IfcLabel data type, is required for correctness checking.
This example represents the use of the PCI-053 by IfcWall.</td></tr>
<tr><td>Checking types</td><td>Value-A, Value-S</td></tr>
<tr><td>Concept</td><td>PCI-053 Building Element Attributes
IfcBuildingElement
<table>
<tr><th>Attributes</th><th>Implementation agreements</th></tr>
<tr><td>GUID</td><td>Must be provided</td></tr>
<tr><td>OwnerHistory</td><td>Must be provided, but may contain dummy data</td></tr>
<tr><td>Name</td><td>Optional</td></tr>
<tr><td>Description</td><td>Optional</td></tr>
<tr><td>ObjectType</td><td>The instance types of IfcBuildingElement in this case should be assigned according to the Appendix A.</td></tr>
<tr><td>ObjectPlacement</td><td>Should carry the location of the precast piece. See different placement methods</td></tr>
<tr><td>Representation</td><td>Should carry the geometric representation of the precast piece.</td></tr>
<tr><td>Tag</td><td>Should carry the Piece Mark of the precast piece (e.g. TS_4201).</td></tr>
</table></td></tr>
</table>

Table 3 continued

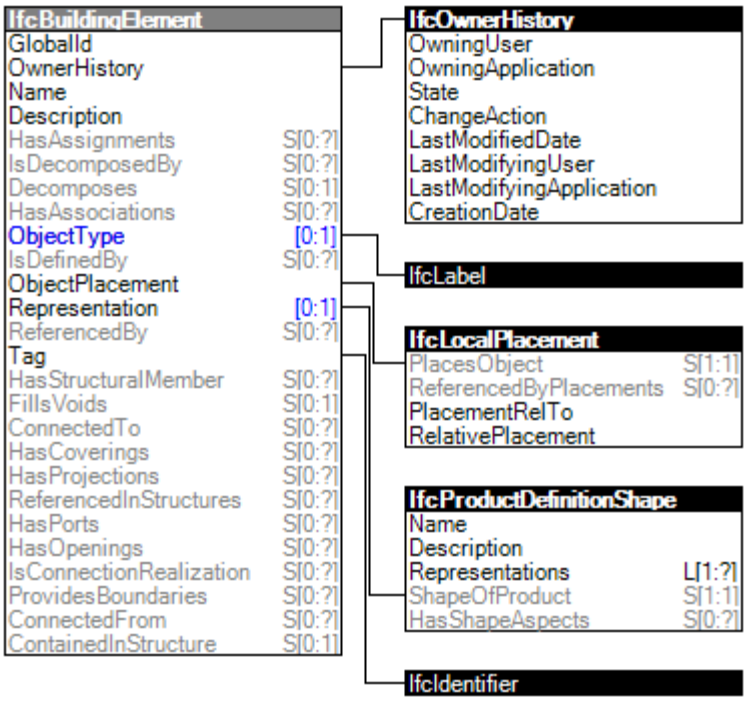
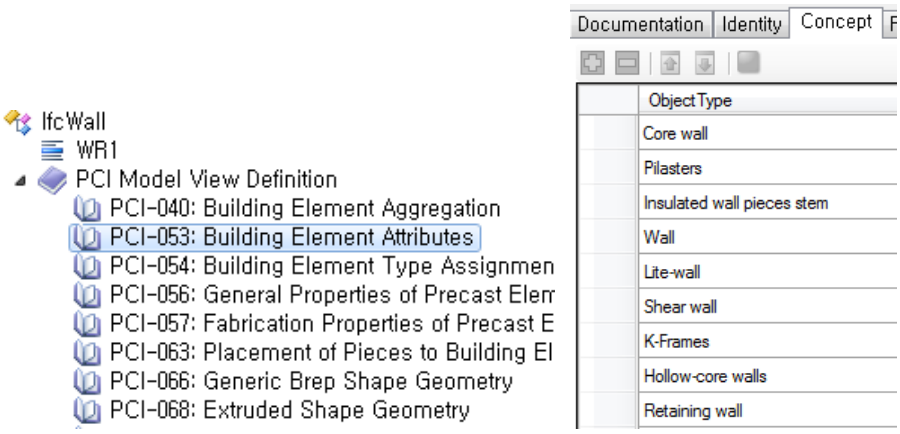
IfcDoc	
Rule definition	

Table 4 The correctness of a simple data value with regard to predefined values

<table>
<tr><td>Requirements</td><td>The concept PCI-107, which represents various attributes of a reinforcing bar, has the attribute, SteelGrade. The SteelGrade attribute requires that its value satisfies one of predefined enumeration values of steel standards defined in the IfcLabel data type. In addition, the BarRole attribute is defined by enumeration of standard types for the role, purpose or usage of the bar. Among ten bar roles, this concept requires an IFC instance to satisfy one of four roles: MAIN, RING, LIGATURE, and NOTDEFINED.</td></tr>
<tr><td>Checking types</td><td>Value-A, Value-S</td></tr>
<tr><td>Concept</td><td>PCI-107 Reinforcing Bar Attribute
IfcReinforcingBar
<table>
<tr><th>Attributes</th><th>Implementation agreements</th></tr>
<tr><td>GUID</td><td>Required</td></tr>
<tr><td>OwnerHistory</td><td>Required</td></tr>
<tr><td>SteelGrade</td><td>Required; The nominal steel grade defined according to local standards.</td></tr>
<tr><td>NominalDiameter</td><td>Required, the cross-section size of the reinforcing bar.</td></tr>
<tr><td>CrossSectionArea</td><td>Required.</td></tr>
<tr><td>Barlength</td><td>Required. The total length of the reinforcing bar. The total length of bended bars are calculated according to local standards with corrections for the bends.</td></tr>
<tr><td>BarRole</td><td>Longitudinal rebar must have “MAIN”
Transvers rebar: must have “RING” or “LIGATURE’
General reinforcing must have “NOTDEFINED”</td></tr>
<tr><td>BarSurface</td><td>Optional; Indicating whether the bar is plain or textured</td></tr>
</table></td></tr>
</table>

Table 4 continued

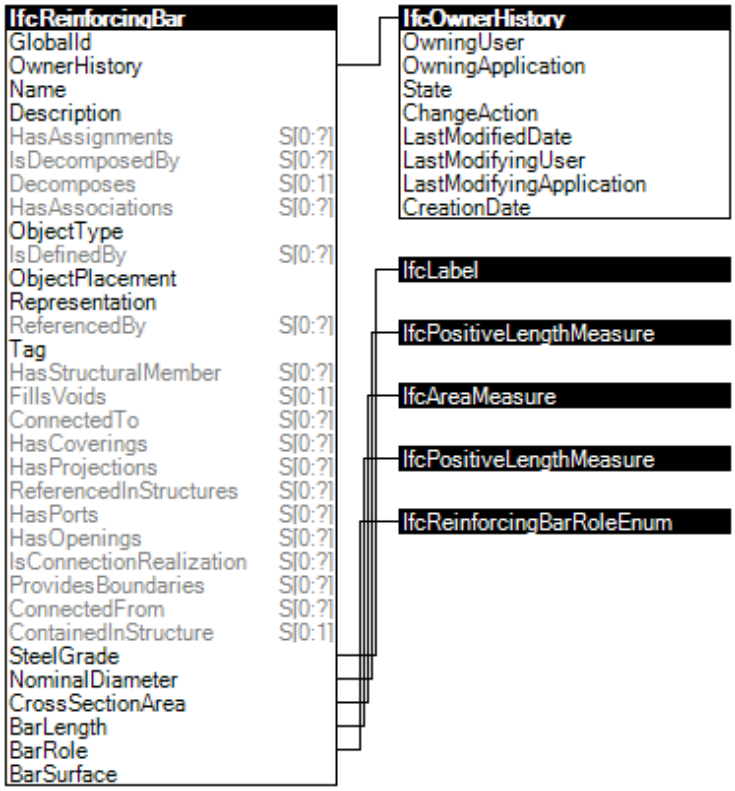
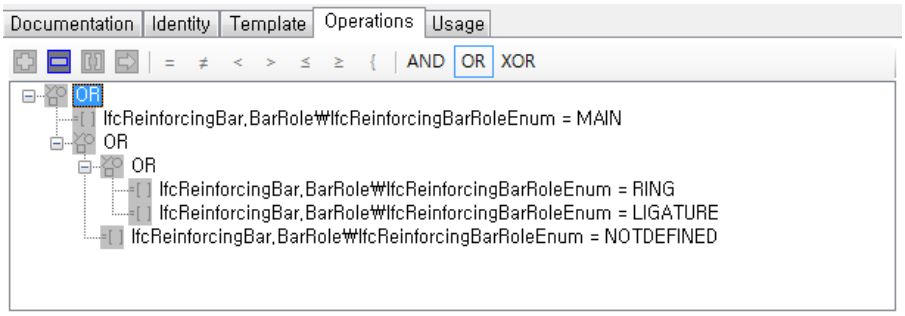
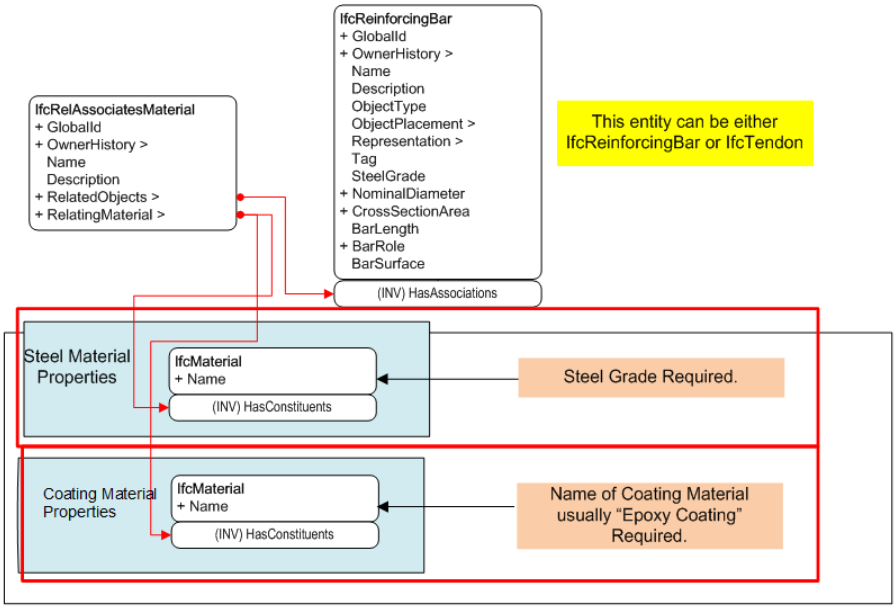
IfcDoc	
Rule Definition	

Table 5 The correctness of a simple data value with regard to an enumeration

<table>
<tr><td>Requirements</td><td>The concept PCI-109, which packages diverse reinforcing element attributes required for design and fabrication, has the attribute type, SteelGrade. This attribute is defined by enumeration of standard types for bars. Several steel standards such as AISC and European steel were added in enumeration values.</td></tr>
<tr><td>Checking types</td><td>Value-A, Value-S, Value-C</td></tr>
<tr><td>Concept</td><td>PCI-109 Reinforcing Bar Attribute
 

IfcRelAssociatesMaterial
<table>
<tr><th>Attributes</th><th>Implementation agreements</th></tr>
<tr><td>RelatedObjects</td><td>List of Objects (reinforcing) with this coating</td></tr>
<tr><td>RelatingMaterial</td><td>Must be a single material</td></tr>
<tr><td>Name</td><td>Must be provided and must contain the text “Reinforcing Steel” or “Tendon Material” or “Rebar Coating Material”</td></tr>
</table></td></tr>
</table>

Table 5 continued

<table>
<tr>
<td>Concept</td>
<td>IfcMaterial
<table>
<tr><th>Attributes</th><th>Implementation agreements</th></tr>
<tr><td>Name</td><td>Must be provided and

If Name = Reinforcing Steel, then must define the steel grade used with a name that refers to CSRI (Concrete Steel Reinforcing Institute) or other standards;

If Name = Tendon Material, then must define tendon material with CRSI or other standards

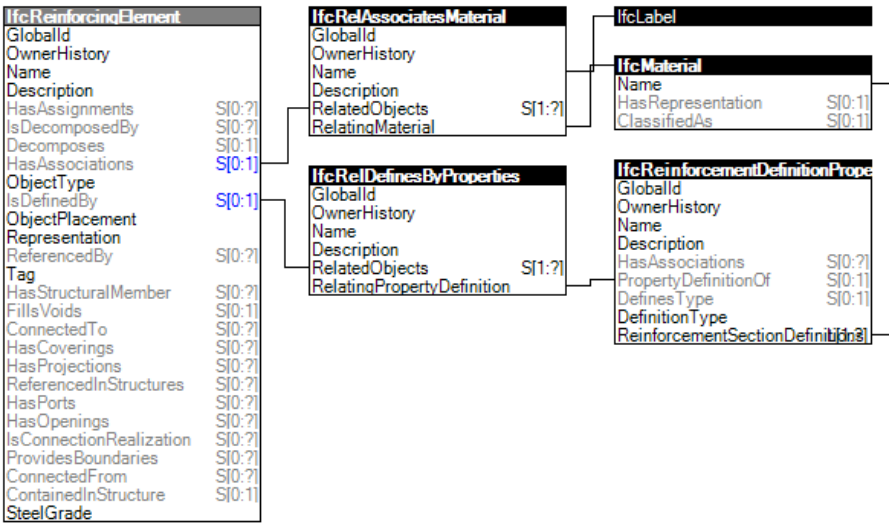
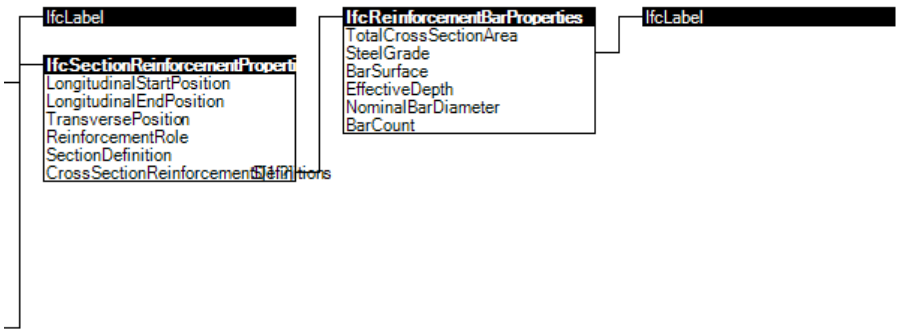
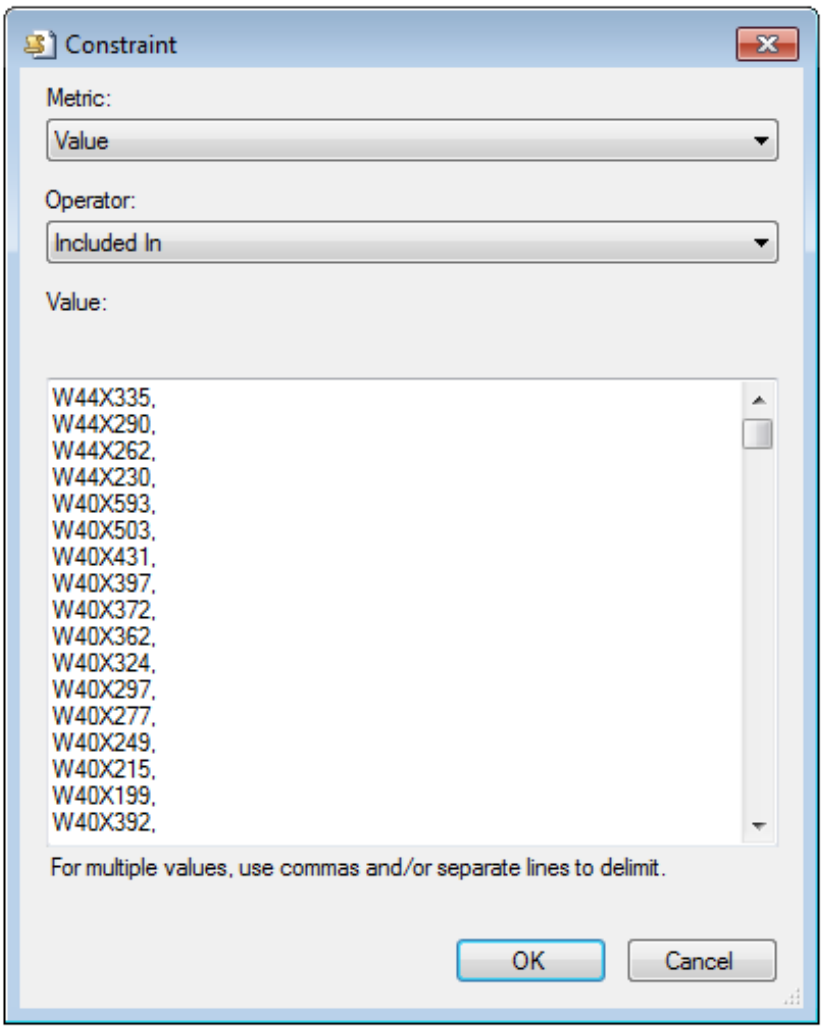
If Name.= Rebar Coating Material, then name should be a recognized coating materiel such as “epoxy coating”</td></tr>
</table></td>
</tr>
<tr>
<td>IfcDoc</td>
<td> 
</td>
</tr>
</table>

Table 5 continued

IfcDoc	
Rule Definition	

4.2.2 Cardinality

The cardinality feature helps restrict the number of values and references of an attribute. Since an MVD following the requirements of the IFC schema has complex hierarchy and inheritances, multiple references and diverse values can be referred and used for relationships and instance values. These specifications require that relations and values satisfy the predefined or user-defined number of attributes, the upper and lower bound. Typically, lower and upper bound of an attribute are predefined in the IFC specifications. Aggregation data types of cardinality at the schema level involve Set, List, and Array. Within the range of cardinality, users add one layer that limits lower and upper bounds.

Table 6 The example of cardinality checking (Lee, Eastman et al. 2015)

Requirements	The PCI-142 concept declares the specifications for the extents of a seam connection for precast pieces. This concept has the Points attribute of IfcPolyline to represent the curve line of a connection. The values of this attribute must satisfy the lower bound of attributes, two points as defined in the IFC schema, LIST [2:?] OF IfcCartesianPoint.
Checking types	Type-A

Table 6 continued

<table>
<tr>
<td>Concept</td>
<td>PCI-142 Precast Seam Connection Location

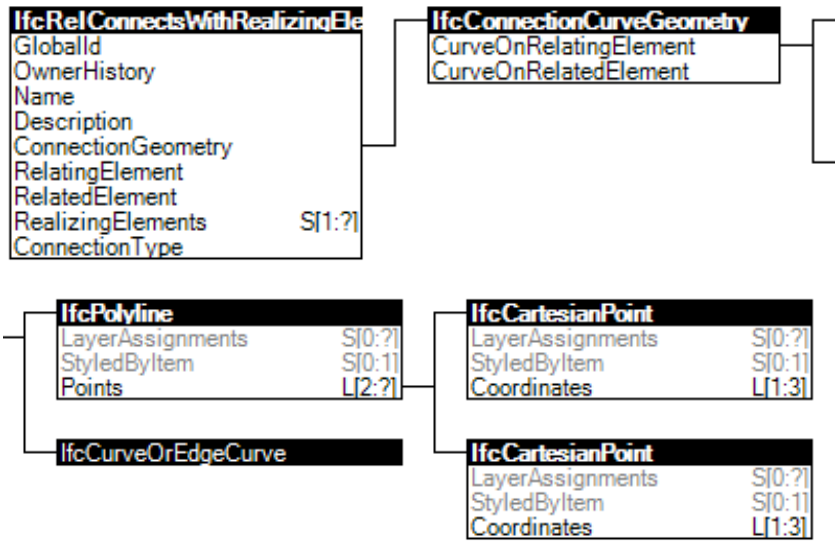
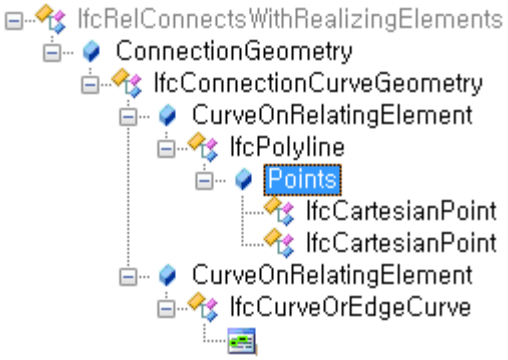
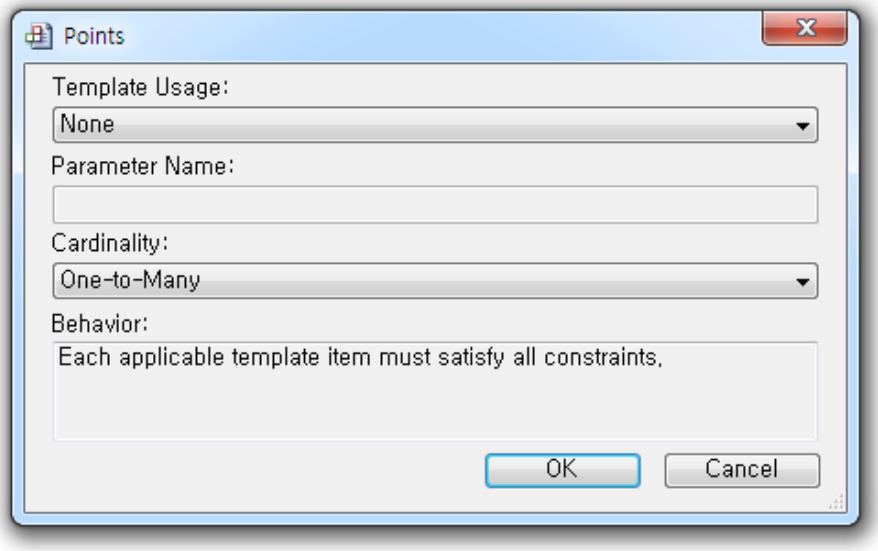
IfcPolyline
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>Points</td><td>Must be at least two points in the local coordinate system of a relating element. A seam connection on a curve will be given by a faceted line; ideally, the points along the line corresponds to the exact location of the connection hardware</td></tr>
</table>
</td>
</tr>
<tr>
<td>IfcDoc</td>
<td> 
</td>
</tr>
</table>

Table 6 continued

Rule Definition	 

Cardinality can be used to define rules of existence and null. With regard to existence, a lower bound should be declared more than one. For null checking, if an upper bound is zero, no value or relation must be added to an attribute. If a lower bound is zero (most inverse relationship has zero as a lower bound), this setting makes it equivalent to optional. As shown in Table 7, the PCI model view declares that an IFC instance file must have values for the ObjectType and Tag attributes.

Table 7 The concept example of existence checking

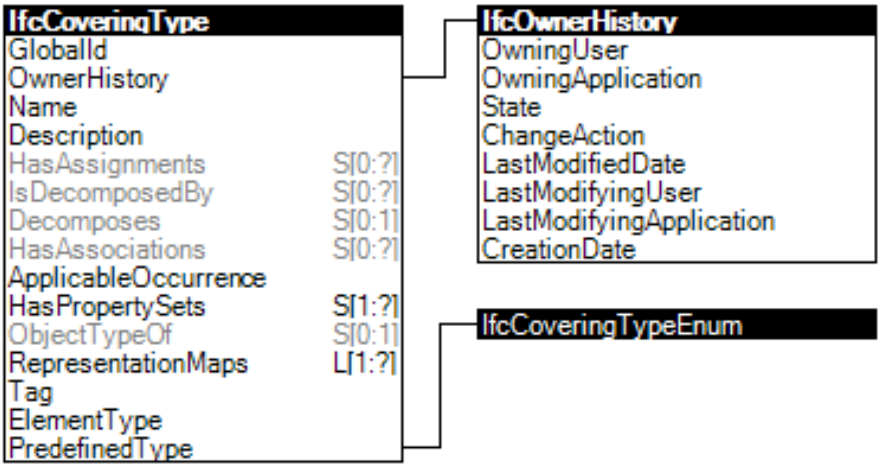
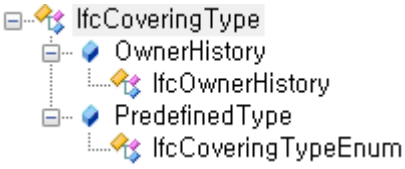
<table>
<tr><td>Requirements</td><td>The PCI-173 concept defines the relationship between a precast finish parch and its attributes. This concept requires that an IFC instance file satisfies an attribute for PredefinedType. Thus, this attribute is validated regarding its existence.</td></tr>
<tr><td>Checking types</td><td>Type-A</td></tr>
<tr><td>Concept</td><td>PCI-173 Precast Surface Treatment Attributes
IfcCoveringType
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>HasPropertySets</td><td>The property set must define the treatment attributes.</td></tr>
<tr><td>RepresentationMaps</td><td>Optional if representation is required at the type level.</td></tr>
<tr><td>PredefinedType</td><td>Should be enumerated by using \IfcCoveringTypeEnum</td></tr>
</table></td></tr>
<tr><td>IfcDoc</td><td> 
</td></tr>
</table>

Table 7 continued

Rule Definition	

An MVD can define that an attribute of an IFC instance file must be null in a certain data exchange. Some values must not be used nor does it require its exchange for a particular domain. Thus, null value checking requires that a value is left blank as the zero arity (Lee, Eastman et al. 2015).

Table 8 The concept example of the null checking

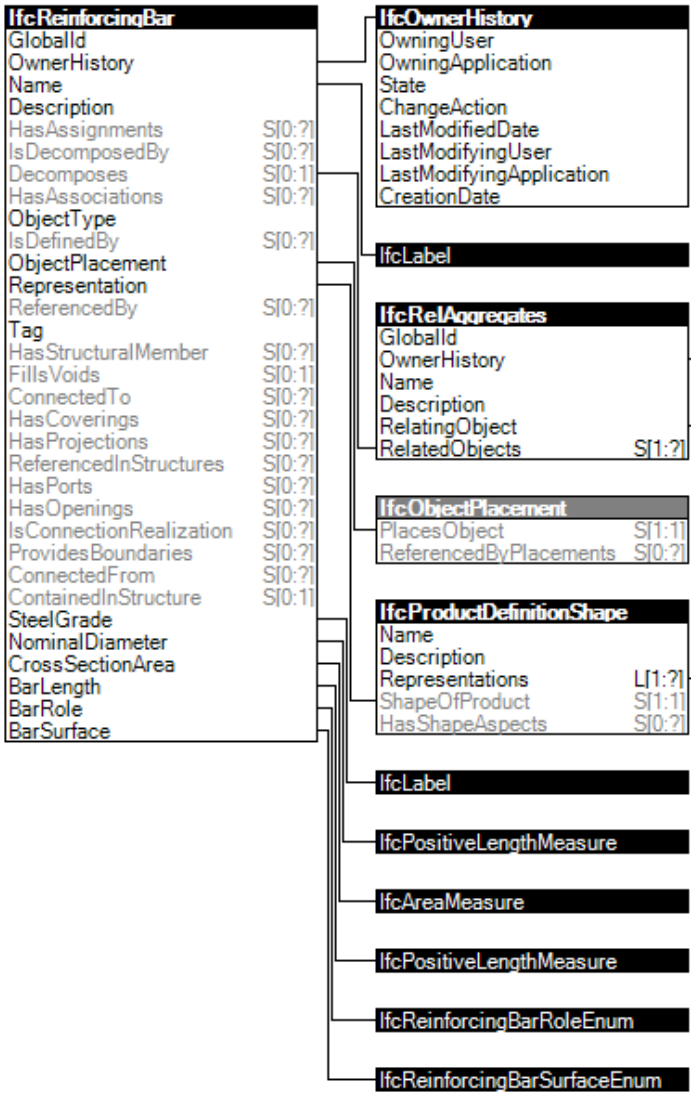
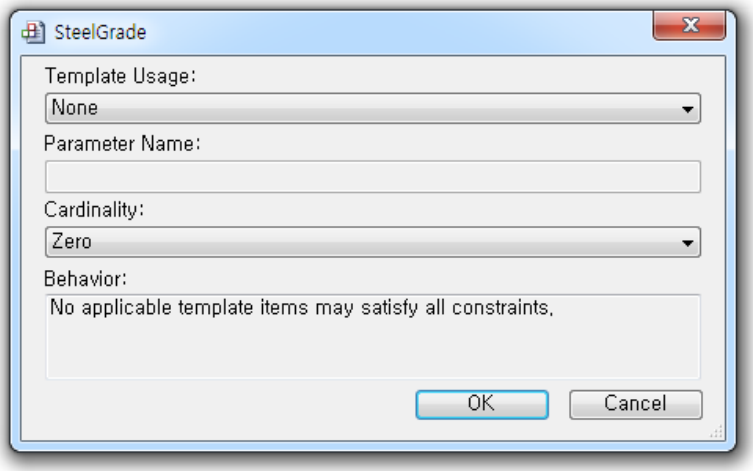
Requirements	The PCI-072 describing the requirements of IfcReinforcingBar has three attributes: SteelGrade, NominalDiameter, and CrossSectionArea. Because such attributes are not used in exchanges by the precast concrete domain, an IFC instance file must have null for three attributes.
Checking types	Value-E, Type-A

Table 8 continued

Concept	PCI-072 Precast Surface Treatment Attributes IfcCoveringType

Attribute	Implementation agreements
ObjectPlacement	Must be provided. Should be a subtype of IfcObjectPlacement
Representation	Must be provided only if there is no type defined. Should be a subtype of IfcProductRepresentation implemented using extruded IfcSweptDiskSolid.
SteelGrade	***NULL. If needed, the nominal steel grade defined according to local standards can be provided as a property set at type level, if type is present.***
NominalDiameter	***NULL. The cross-section size of the reinforcing bar must be attached as a property set at type level, if type is present or else at the individual bar level.***
CrossSectionArea	***NULL. Should be attached as a property set at type level, if type is present.***
Barlength	***NULL. The total length of the reinforcing bar. The total length of bended bars are calculated according to local standards with corrections for the bends. Should be attached as a property set at type level. Rebar of same type are interchangeable.***
BarRole	***Must be .NOTDEFINED.***
BarSurface	***NULL .Indicating whether the bar is plain or textured. This should also be attached as a property set at type level, if type is present.***

Table 8 continued

IfcDoc	
Rule Definition	

4.2.3 Uniqueness

Uniqueness checking is required for data exchange in that entities can be classified and labeled by a unique identifier. The IFC schema defines an UNIQUE rule and the MVD also can include a unique constraint such as a unique value for a TAG attribute. As discussed in Section 3, uniqueness of attribute values can be validated either globally or locally. Global uniqueness refers that the value of an attribute must be unique in all instances of a P21 file. Local uniqueness is defined by data type defined in the schema specifications and is implemented for checking values embedded in a given aggregation data according to uniqueness.

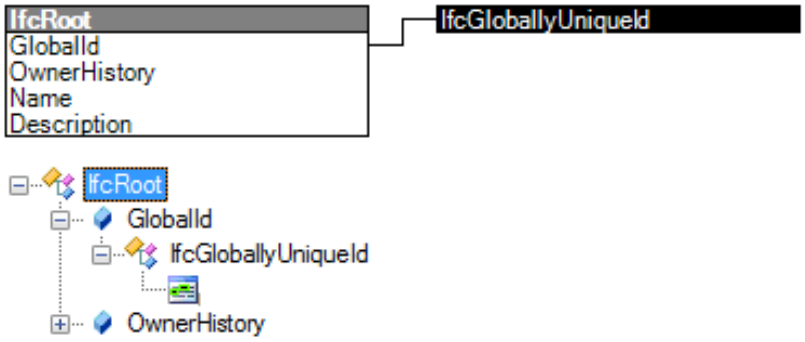
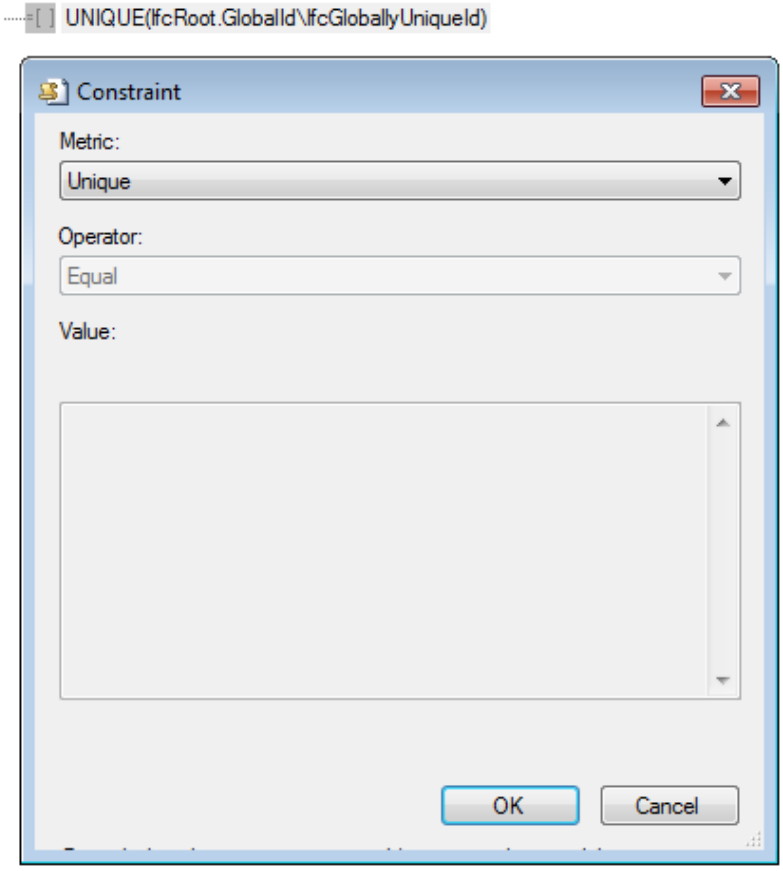
For global uniqueness checking, because GUID of all entities must be unique, it can be validated by a single concept having a root entity as IfcRoot. In addition, the concept can be assigned to IfcRoot. This individual concept helps avoid causing an error when an entity is referred by multiple instances. In developing uniqueness checking, when one instance is referred by multiple instances, validation of uniqueness checking shows FAIL because several references share the same values of the instance. In the implementation level, it was a logically expected behavior because uniqueness evaluates that a value traversed from a graph occurs only once. Thus, sharing the same entity results in being traversed multiple times, representing FAIL for all duplicates without the first unique instance. To ameliorate this technical challenge, this dissertation used a separate concept template that evaluate the GlobalId values of IFC instance files regarding uniqueness across all instances deriving from IfcRoot, then define a rule that applies directly to IfcRoot.GlobalId. This path helps skip possibility for any graph to reach the same value

from multiple paths. GUID require 22 length string, which must be unique within all instances. Also, validation is case-sensitive as the IFC specification follows. In terms of TAG attribute, it can be validated by a single concept having a root entity as IfcRoot. This concept has to be assigned to each entity that requires this validation type.

Table 9 The concept example of the global uniqueness of a value with regard to the range of an entire instance file and one of the internal uniqueness of a value with regard to one specific attribute

<table>
<tr><td>Requirements</td><td>The MVC-581 defines general attributes of the IfcRoot entity including GlobalId, OwnerHistory, Name, and Description.</td></tr>
<tr><td>Checking types</td><td>Value-U</td></tr>
<tr><td>Concept</td><td>MVC-581 Root Attributes
IfcRoot
<table><tr><th>Attribute</th><th>Implementation agreements</th></tr><tr><td>GlobalId</td><td>Definition from IAI: Holds an identifier that is unique throughout the software world. This is also known as a Globally Unique Identifier (GUID) or Universal Unique Identifier (UUID) by the Open Group. The identifier is generated using an algorithm published by the Object Management Group. The algorithm is explained at the open group website. The Microsoft Foundation Class (MFC) function "CoCreateGuid", which is an implementation of the above algorithm, has been used by many IFC implementers to create an identifier.</td></tr><tr><td>OwnerHistory</td><td>Must be provided.</td></tr><tr><td>Name</td><td>NULL</td></tr><tr><td>Description</td><td>NULL</td></tr></table></td></tr>
</table>

Table 9 continued

IfcDoc	
Rule Definition	

With regard to local uniqueness, it is defined by a data type. SET aggregation data types requires that a value must be unique within the corresponding aggregation.

Table 10 Uniqueness of aggregation data types (Lee, Eastman et al. 2015)

<table>
<tr><td>Requirements</td><td>The PCI-098 has requirements for the representation of the embedded type assignment. Using IfcDiscreteAccessoryType, this concept describes RepresentationType for the identification of a representation. Since the origin of its mapping is defined by the representation item, the values of IfcRepresentationMap should be unique in the aggregation LIST type. This local uniqueness checking evaluates each value of the aggregation data type according to uniqueness using only associated values.</td></tr>
<tr><td>Checking types</td><td>Value-U, Type-A</td></tr>
<tr><td>Concept</td><td>IfcDiscreteAccessoryType
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>HasPropertySets</td><td>Optional. Should use IfcPropertySetDefinition</td></tr>
<tr><td>Representation Maps</td><td>Optional. Should use list of unique IfcRepresentationMap</td></tr>
<tr><td>Tag</td><td>The tag should be an unique and company specific label.</td></tr>
<tr><td>ElementType</td><td>Optional Label</td></tr>
</table></td></tr>
</table>

Table 10 continued

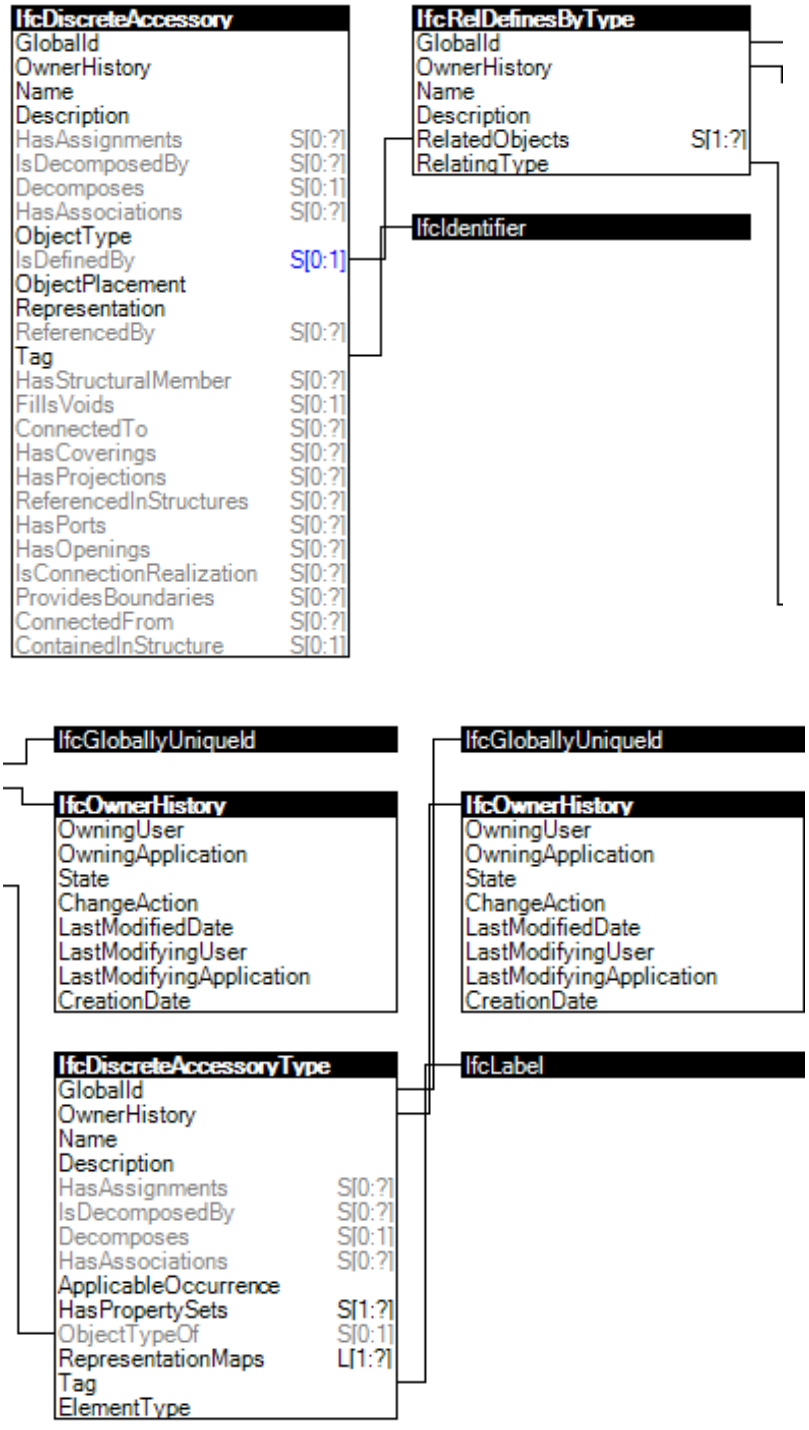
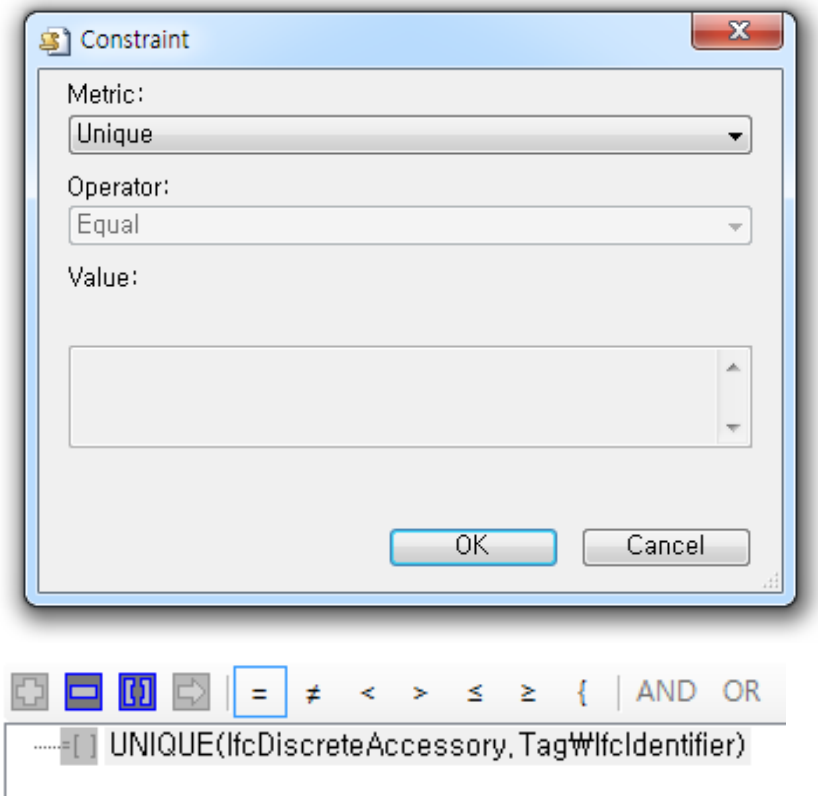
IfcDoc	

Table 10 continued

Rule Definition	

4.2.4 Reference/Inverse relationship

An attribute requires particular types of entities and relationships. This requirement is generated by drawing a relational structure. If an attribute is connected to another entity, an instance is validated with regard to accuracy of a reference. An inverse relationship is also defined for reference validation.

Table 11 Reference relationship

Requirements	The PCI-054 regarding Building Element Type Assignment describes associates an ElementType to an Element instance, associating the ElementType's information.
Checking types	Type-R, Type-I, Type-S

Table 11 continued

<table>
<tr>
<td>Concept</td>
<td>PCI-054 Building Element Type Assignment

IfcRelDefinesByType
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>RelatedObjects</td><td>If the IfcDiscreteAccessory is assigned to an IfcBuildingElement, and this element defines its local placement, then the placementRelTo relationship of IfcLocalPlacement shall point to the local placement of the IfcBuildingElement. Is a Set. Must be subtype of IfcBuildingElement. See precast piece or precast piece mark concept bindings for rules about appropriate subtype selection. May be: IfcBuildingElementProxy, IfcCovering, IfcBeam, IfcColumn, IfcCurtainWall, IfcDoor, IfcMember, IfcRailing, IfcRamp, IfcRampFlight, IfcWall, IfcSlab, IfcStairFlight, IfcWindow, IfcStair, IfcRoof, IfcPile, IfcFooting, IfcBuildingElementComponent, IfcPlate IfcElementAssembly</td></tr>
<tr><td>RelatingType</td><td>Must be a single sub-type of IfcBuildingElementType. These are: IfcCoveringType, IfcBeamType, IfcMemberType, IfcColumnType, IfcWallType, IfcSlabType, IfcStairFlightType, IfcRampFlightType, IfcCurtainWallType, IfcRailingType, IfcBuildingElementProxyType, IfcPlateType The subtype must match the subtype of IfcBuildingElement (e.g. IfcBeam must have IfcBeamType for RelatingType)</td></tr>
</table>
</td>
</tr>
</table>

Table 11 continued

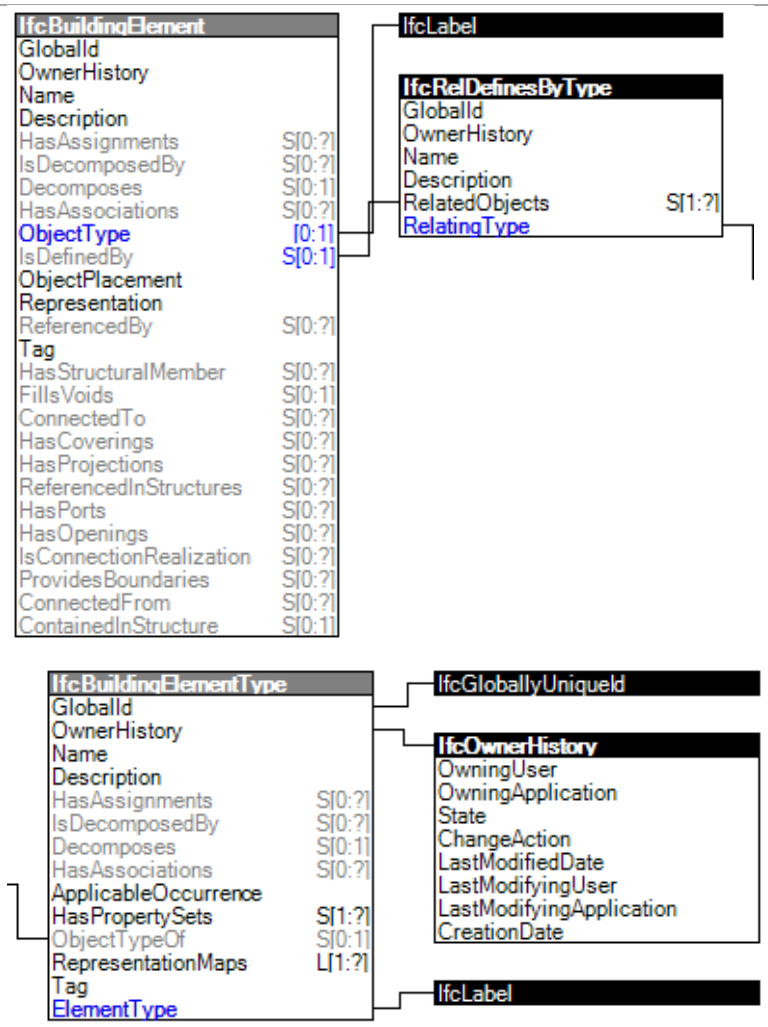
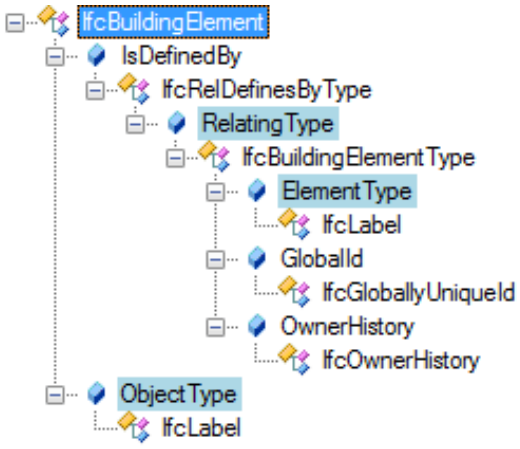
IfcDoc	
Rule Definition	

Table 12 One to many relationship

<table>
<tr><td>Requirements</td><td>The PCI-070 should satisfy one of two entities: IfcPolyline or IfcTrimmedCurve. Two entities for the ParentCurve attribute can be added and its cardinality is set as Zero-To-One.</td></tr>
<tr><td>Checking types</td><td>Value-C, Type-A, Type-C</td></tr>
<tr><td>Concept</td><td>MVC-581 Root Attributes
IfcCompositeCurveSegment
<table><tr><th>Attribute</th><th>Implementation agreements</th></tr><tr><td>Transition</td><td>Defines transition type from this segment to next on list. Must be one of enumeration of IfcTransitionCode; Must not be .DISCONTINUOUS</td></tr><tr><td>SameSense</td><td>BOOLEAN whether segment sense of direction is same as parent curve</td></tr><tr><td>ParentCurve</td><td>References profile curve;
If IfcProductDefinitionShape.description = “appx shape”, then this should be a single IfcPolyline;
If IfcProductDefinitionShape.description = “detail shape”, then this can be any mixture of IfcPolyline and IfcTrimmedCurve;</td></tr></table></td></tr>
</table>

Table 12 continued

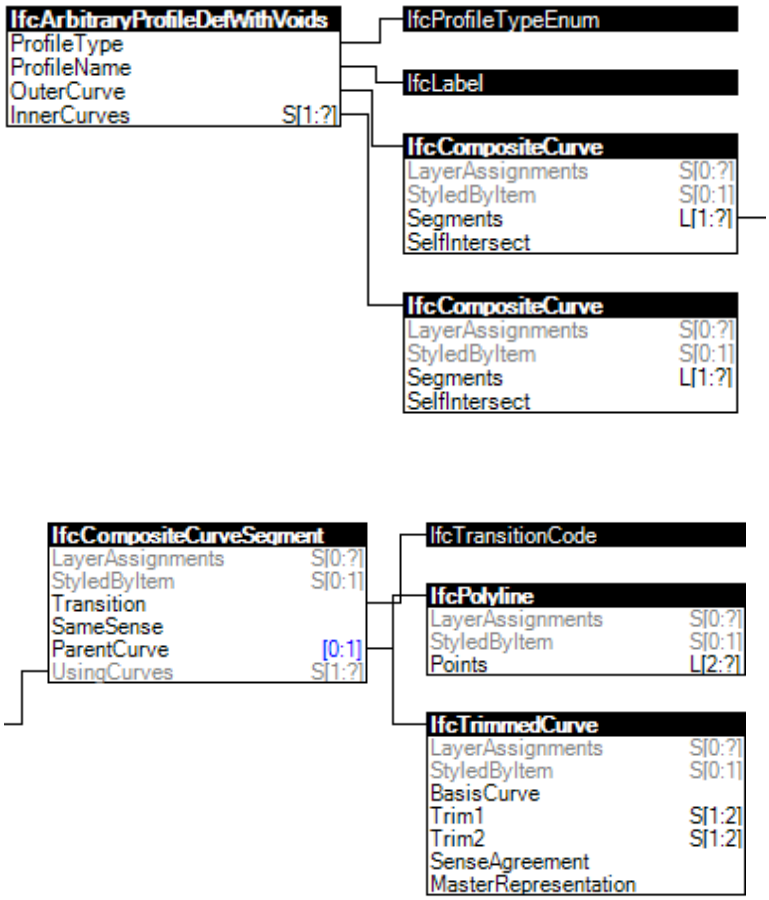
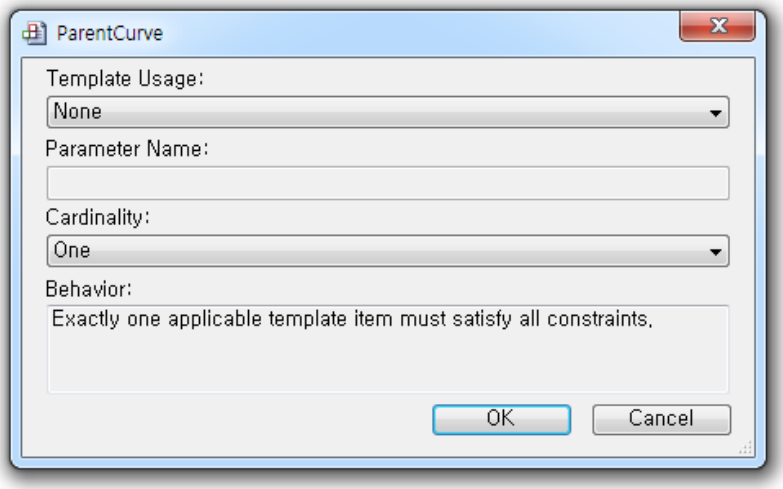
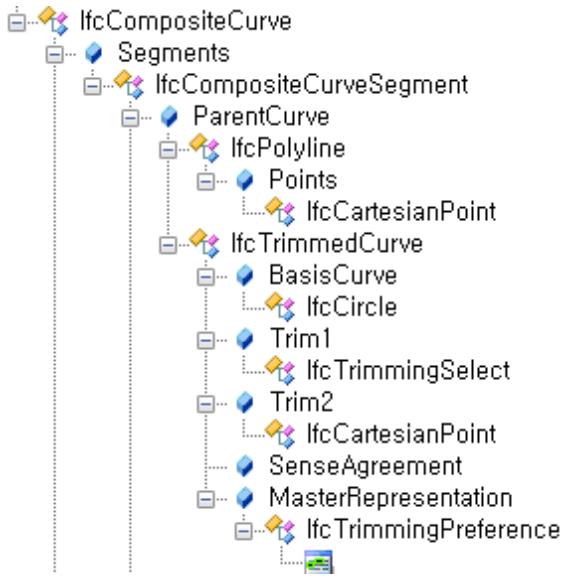
IfcDoc	
Rule Definition	

Table 12 continued

Rule Definition	

4.2.5 Relations based on types of objects and aggregation

The IFC schema is open to diverse interpretations so that native objects can be represented in various ways. The objective of the IFC specifications is to translate native objects into a neutral format and transfer design data for diverse domain professionals. Thus, even the same object that involves the same design data can be represented distinctly in accordance with BIM authoring tools that domain experts use. In other words, data exchange requirements should be applied based on types of objects and relations.

The PCI-103 concept is intended to validate the RelAggregation between ElementAssemblies. For this case, validation should differentiate type of relations and validate only the relevant entities. Types of relations and objects should be distinguished by a specific identifier indicating applicable entities. One suggested method to identify

type of the relationship was to use the Name attribute of the IfcRelAggegates, which can be used as a parameter to identify type of relationship. For example, if the PCI-103 concept can be implemented in validation only when the Name of the IfcRelAggregates in the instance file is ElementAssembly, this condition would be able to preclude users from checking the diverse types of relationship such as those associated with IfcColumn. In other words, if the Name has Null or a various other value, the IfcDoc does not execute the subordinate checking processes associated with RelatedObjects, but shows NOT EXECUTED. A report might also represent NOT APPLICABLE, NOT RELATED, or NOT VALIDATED.

This feature might be capable to validate various PropertySet of one entity. For example, IfcColumn can have two types of PropertySet named Precast General Attribute and Precast Fabrication Attribute. Since the current validation application cannot identify types of PropertySet, two concepts are executed for validating two PropertySets of an instance file iteratively, which causes FAIL for two concepts. However, requiring specific usage of Name at IfcRelAggregates, as such relationship objects, is typically not exposed in native applications – there would be no place to define such naming in typical applications. Thus, this checking type can be accomplished by defining a conditional parameter, and then indicating the condition at the template. For example, the built-in template for property set checking is executed depending on the PredefinedType value of IFC instance files.

Table 13 Relationship based on types of object

Requirements	The PCI-103 concept assumes another concept PCI- 072 aggregates individual reinforcing elements into IfcElementAssembly for reinforcement units, such as stirrups for column rebar cage. This concept supports, as needed, the aggregation of Multiple ElementAssembly into a higher level ElementAssembly. In this case, individual instances of ElementAssembly at the disaggregated level should not carry the aggregation to the structural element (precast piece); it should be provided through the top level element assembly entity. The PCI-103 concept defines the RelAggregation for two IfcElementAssembly entities. Thus, this concept expects that an instance file consists of two IfcElementAssembly entities for Decompose and IsDecomposedBy inverse relationship. However, since validation is executed based on the root entity, IfcElementAssembly, the entity related to Decomposes is validated for the relation of the IsDecomposedBy, which results in reporting FAIL. Thus, similar to identifying type of object using the Name attribute, IfcElementAssembly can be designated as main and dependent assemblies.
Checking types	Value-C, Type-C, Type-I

Table 13 continued

Concept

PCI-103 Aggregation of Reinforcing Assemblies

IfcElementAssembly (RelatingObject):

Attribute	Implementation agreements
Description	Required; Aggregates reinforcing assemblies: if mesh aggregation, value is “MESH ASSEMBLY”, if rebar aggregation , value is ‘REBAR ASSEMBLY’, if tendon aggregation, value is ‘TENDON ASSEMBLY’
(INV) Decomposes	Decomposes an assembly to its reinforcement element aggregation constituents, which is usually a rebar assembly.
PredefinedType	Required; All are aggregations, not groups

IfcElementAssembly (RelatedObjects):

Attribute	Implementation agreements
Description	If rebar, Aggregation. Tag must be ‘REBAR ASSEMBLY’, if mesh, the Aggregation tag must be ‘MESH ASSEMBLY’;
(INV) Decomposes	Reinforcing Element Aggregations which are aggregated into higher level assemblies which usually is a rebar cage.
PredefinedType	Required; All are aggregations, not groups

Table 13 continued

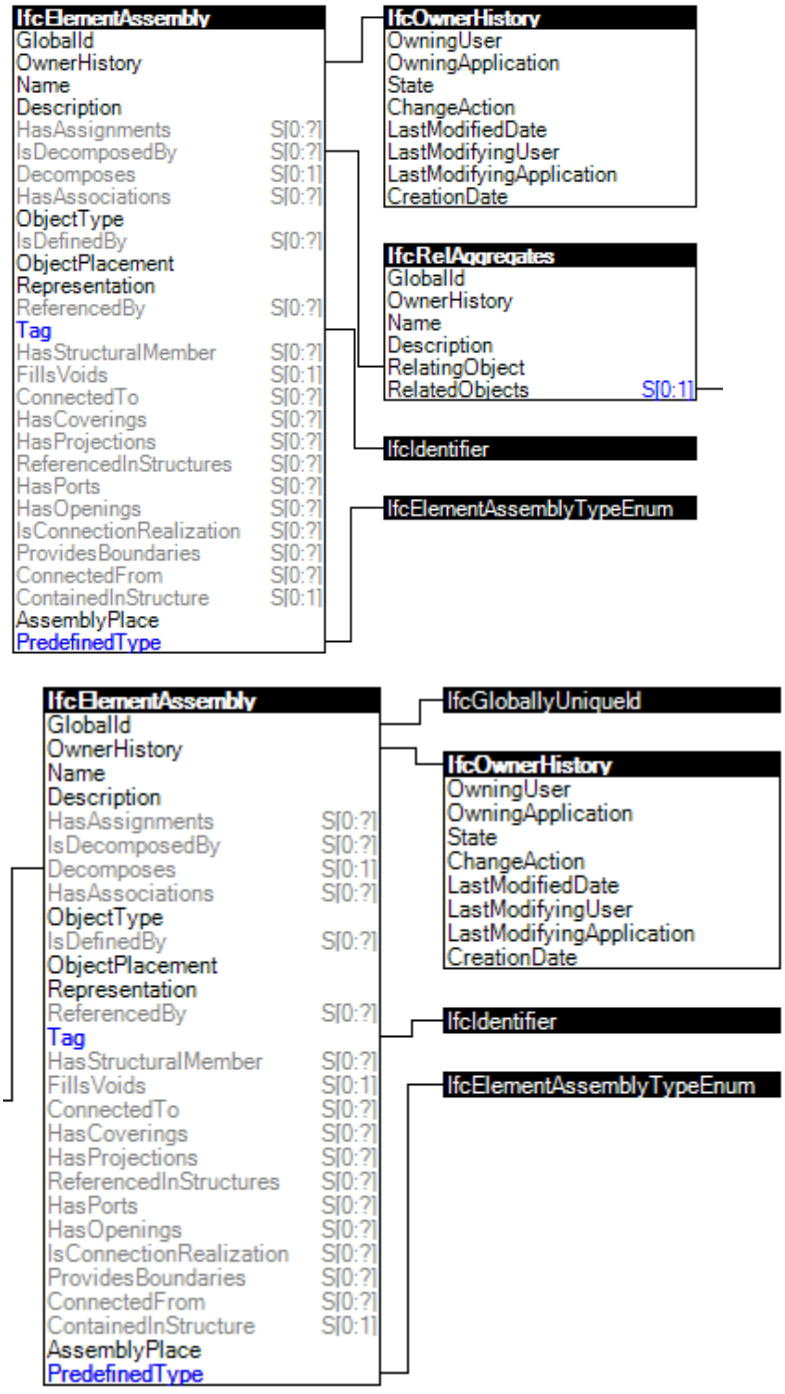
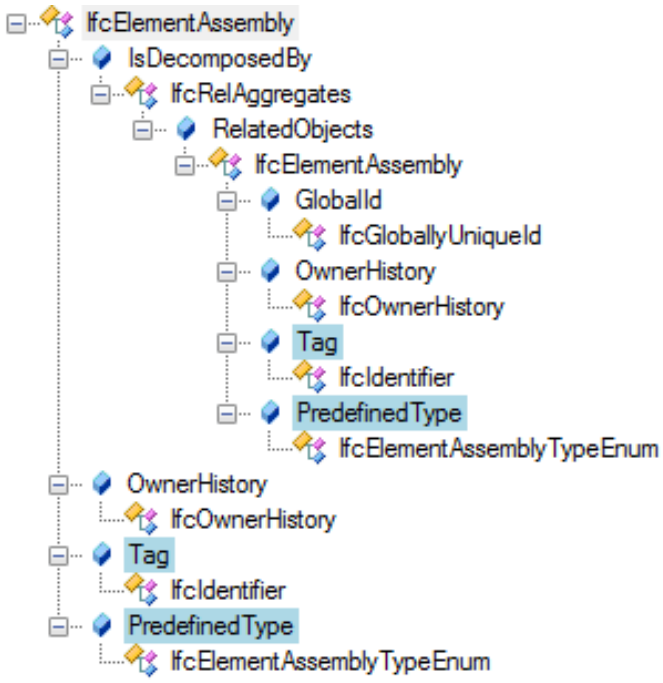
IfcDoc	

Table 13 continued

Rule Definition	

4.2.6 Subtype checking

One object in the MVD sometimes requires several relationships under the same category. For example, a slab, a beam, a column, a wall, a reinforcing element, and other element accessory include similar requirements for ObjectType, Representation, and Tag. To define such requirements, plenty of rules and relationships should be declared in a concept. Such objects, however, exist underneath the IfcBuildingElement entity, and thus, one relationship with a higher-level entity with shareable requirements allows that all subtype objects can be validated according to one concept requirements. Using the inheritance structure of IFC specifications, concept templates can be reused to generate concise exchange requirements.

Table 14 The example of subtype checking

<table>
<tr><td>Requirements</td><td>The PCI-053 concept, which generally specifies the attributes of associated IfcBuildingElement types, is iteratively referred by the subtypes of IfcBuildingElement such as IfcWall. The subtypes referring to this concept must be evaluated by the specifications of this concept. The ObjectType of subtypes of IfcBuildingElement are defined in the document called Appendix A. This document is attached in the Appendix section.</td></tr>
<tr><td>Checking types</td><td>Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-053 Building Element Attributes
IfcBuildingElement (ABS) :
<table>
<tr><th>Entity</th><th>Implementation agreements</th></tr>
<tr><td>ObjectType</td><td>The instance types of IfcBuildingElement in this case should be assigned according to the Appendix A.</td></tr>
<tr><td>ObjectPlacement</td><td>Should carry the location of the precast piece. See different placement methods</td></tr>
<tr><td>Representation</td><td>Should carry the geometric representation of the precast piece.</td></tr>
<tr><td>Tag</td><td>Should carry the Piece Mark of the precast piece (e.g. TS_4201).</td></tr>
</table></td></tr>
</table>

Table 14 continued

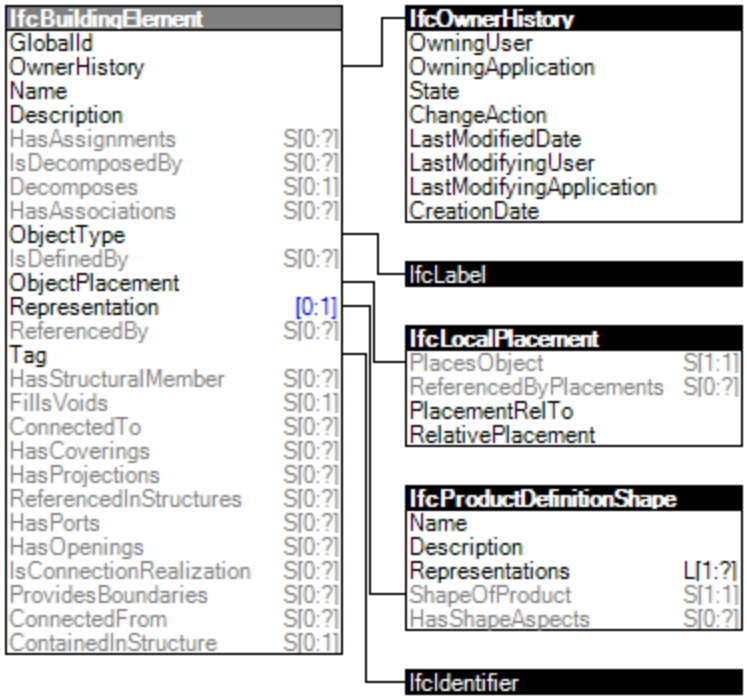
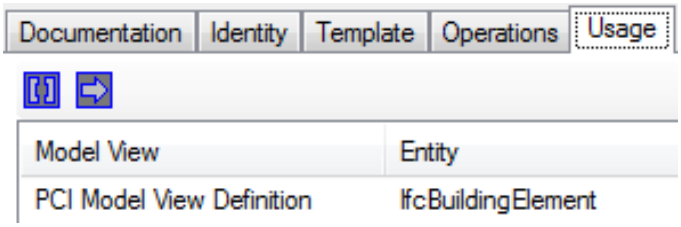
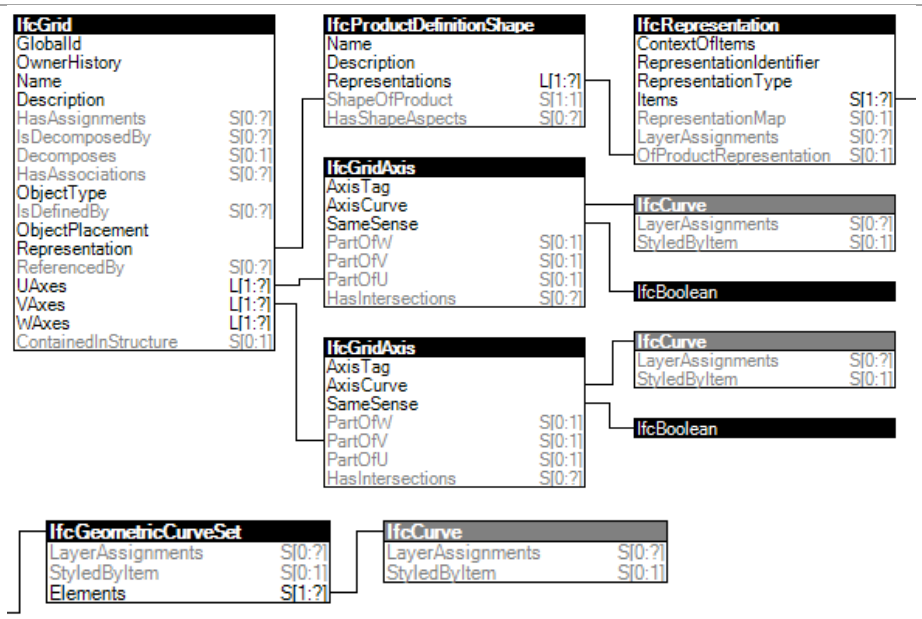
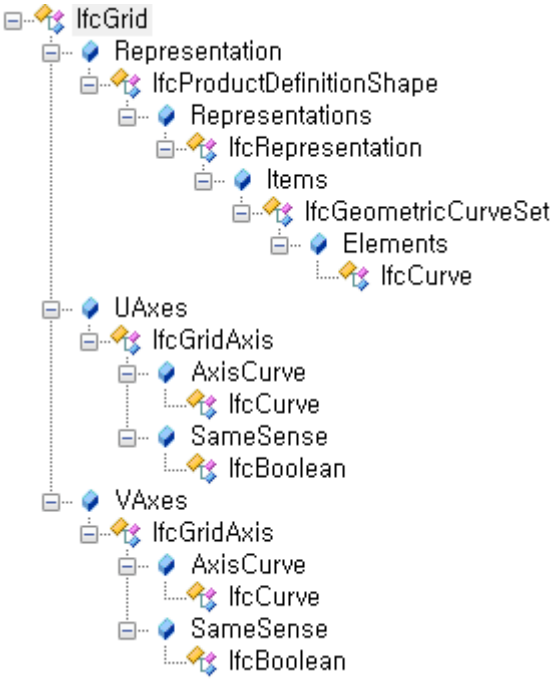
IfcDoc	
Rule Definition	

Table 15 The concept example of reference and subtype checking (Lee, Eastman et al. 2015)

<table>
<tr><td>Requirements</td><td>The PCI-048 concept showing the specifications of IfcGrid for precast concrete declares that the IfcCurve entity must fulfill one of the following types: IfcPolyline, IfcCircle, and IfcTrimmedCurve. Thus, based on corresponding subtypes identified by IfcDoc, users can evaluate an IFC instance file to see whether instances satisfy the listed subtypes.</td></tr>
<tr><td>Checking types</td><td>Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-048 Grid Representation

Grid Representation
<table>
<tr><th>Entity</th><th>Implementation agreements</th></tr>
<tr><td>IfcGeometric CurveSet</td><td>The IfcGeometricCurveSet shall be an Item of the IfcShapeRepresentation. It should contain an IfcGeometricCurveSet containing subtypes of IfcCurve, each representing a grid axis.</td></tr>
<tr><td>IfcCurve</td><td>Applicable subtypes of IfcCurve are: IfcPolyline, IfcCircle, IfcTrimmedCurve (basedonBaseCurve referencing IfcLine or IfcCircle).</td></tr>
<tr><td>IfcGridAxis</td><td>AxisCurve is the underlying curve which provides the geometry for this grid axis. Each instance of IfcGridAxis refers to the same instance of IfcCurve that is contained within the IfcGeometricCurveSet that represents the IfcGrid.</td></tr>
</table></td></tr>
</table>

Table 15 continued

IfcDoc	
Rule Definition	

Despite the benefits of inheritance, some requirements are unnecessarily inherited to subtype entities. Thus, users need to limit inheritance and subtype checking. Suppressing feature helps limit concept rules for lower-level entities.

4.2.7 Conditional checking

Conditional checking means that if a defined condition is satisfied, the corresponding rules are executed to evaluate instances. Otherwise, a validation report represents FAIL or SKIP. Entities, attributes, rules, and concepts can be defined as a conditional relationship, which has conditional requirements and dependent rules on the references of attributes. For example, unit checking can use a conditional feature.

Table 16 The example of the conditional checking (Lee, Eastman et al. 2015)

Requirements	This PCI-093 concept shows the specifications of the surface treatments using the IfcDiscreteAccessory entity. In terms of the ObjectPlacement attribute, its reference should be selectively defined based on the conditions: (1) the assignment of IfcDiscreteAccessory to IfcBuildingElement and (2) the local placement of IfcDiscreteAccessory. The conditional checking is also used in the PCI-103 concept for defining a Tag attribute of IfcElementAssembly. The PCI-103 concept, which represents the specifications of assignments between reinforcing element aggregations, requires distinct values for the Tag attribute according to types of IfcElementAssembly.
Checking types	Value-C, Type-R

Table 16 continued

Concept

PCI-093 Surface Treatments

IfcDiscreteAccessory

Attribute	Implementation agreements
GUID	Must be provided
OwnerHistory	Must be provided, but may contain dummy data
Name	Optional
Description	Optional
ObjectType	Optional
ObjectPlacement	***If the IfcDiscreteAccessory is assigned to an IfcBuildingElement, and this element defines its own LocalPlacement, then the placementRelTo relationship of IfcLocalPlacement shall point (if given) to the local placement of the IfcBuildingElement.***
Representation	The geometric representation of IfcDiscreteAccessory is given by the IfcProductDefinitionShape, allowing multiple geometricrepresentations.
Tag	The tag should be an unique and company specific label.

PCI-103 Reinforcing Element Aggregation Association to Rebar Cage

IfcElementAssembly

Attribute	Implementation agreements
GUID	Must be provided
OwnerHistory	Must be provided, but may contain dummy data
Name	Optional

Table 16 continued

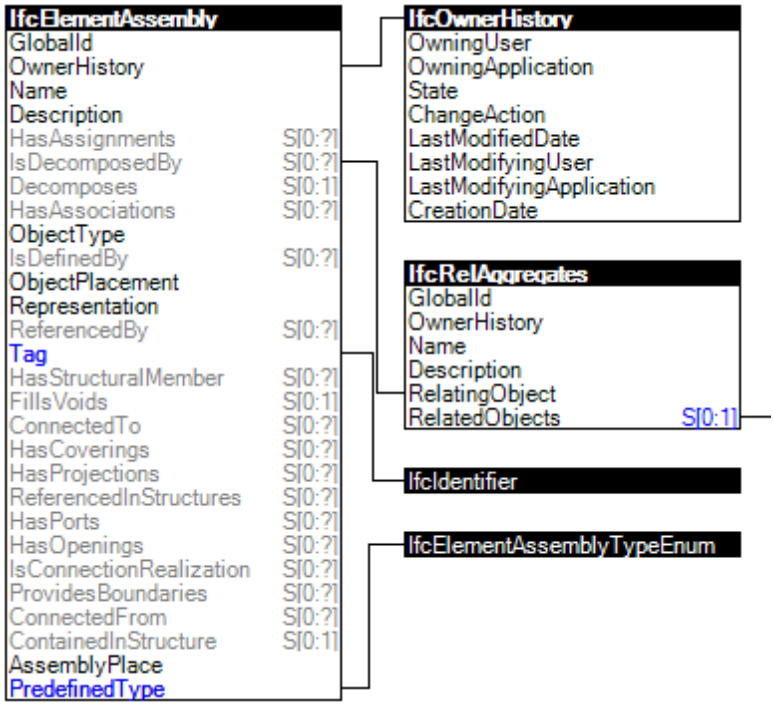
<table>
<tr>
<td>Concept</td>
<td>
<table>
<tr><td>Description</td><td>Optional</td></tr>
<tr><td>Tag</td><td>If we have aggregation of IfcReinforcingBar, it must be the label: “Rebar Assembly“, If we have aggregation of IfcTendon, it must be the label: “Tendon Group“, and If we have aggregation of IfcMesh, it must be the label: “Mesh Group“</td></tr>
<tr><td>(INV) IsDecomposedBy</td><td>Reinforcing Element Aggregations aggregated into higher level assemblies that usually comprise a rebar cage.</td></tr>
</table>
</td>
</tr>
<tr>
<td>IfcDoc</td>
<td> 
</td>
</tr>
</table>

Table 16 continued

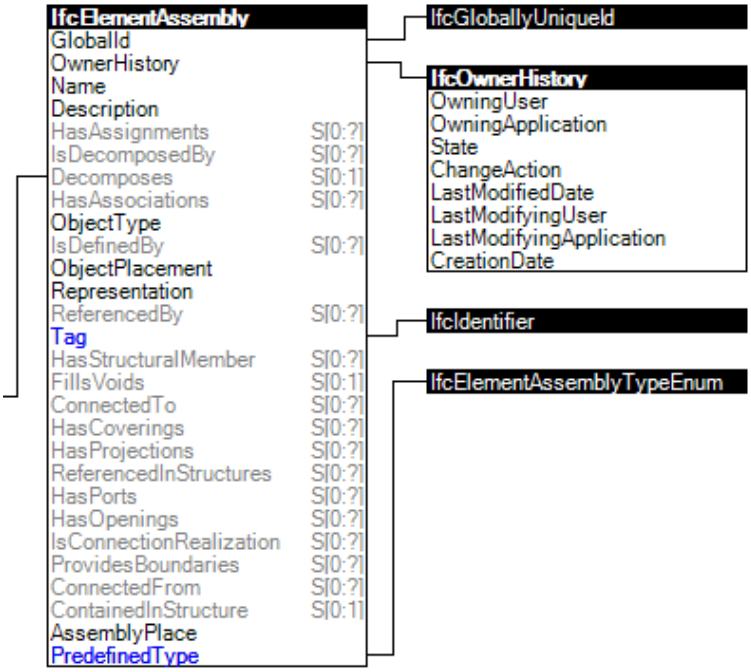
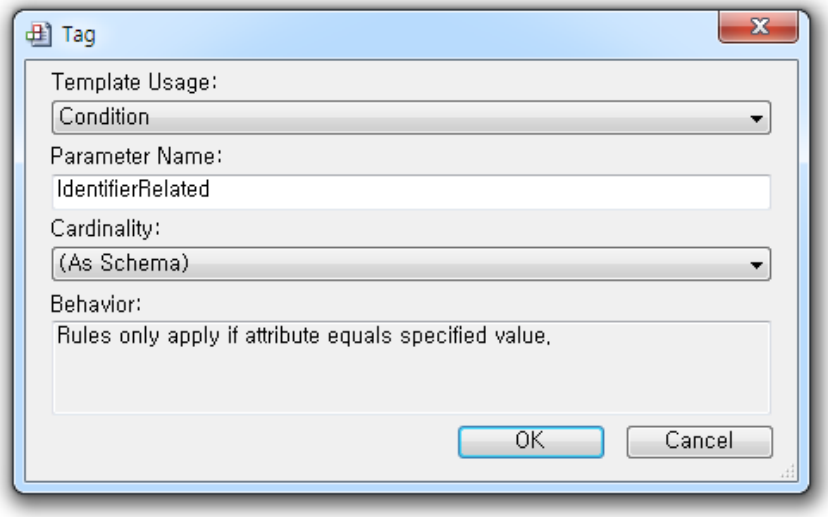
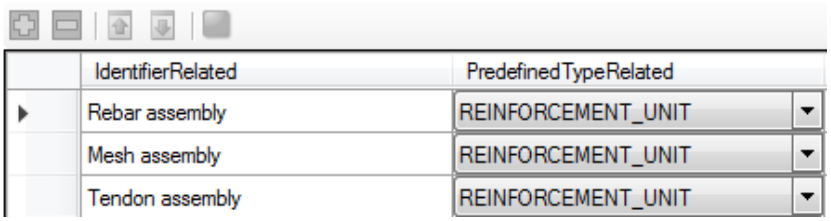
IfcDoc	
Rule Definition	 

Table 16 continued

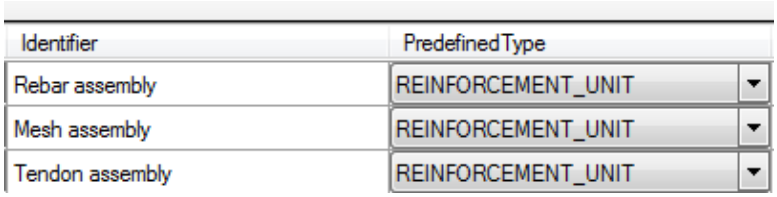
Rule Definition	

Table 17 The concept example of the conditional checking of a value and a type

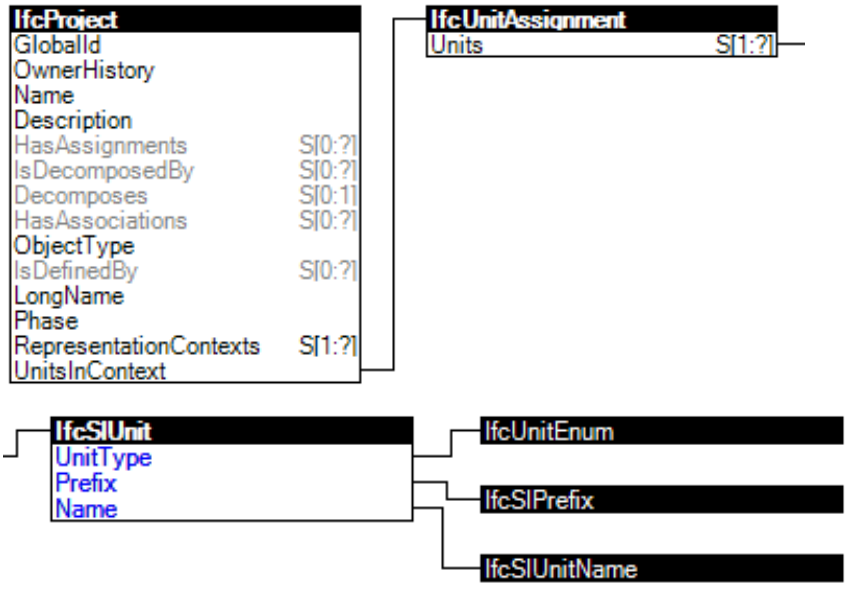
Requirements	The first template is conditional [see '?' notation on UnitType] and has cardinality of the Units attribute set to 1:1. The file includes one LENGTHUNIT with METRE. The file contains exactly one THERMODYNAMICTEMPERATUREUNIT with DEGREES_CELSIUS (not KELVIN), so that fails. The file does not contain any FORCEUNIT, however passes because the rule indicates a condition (* is shown to differentiate passing because data isn't present). The second template is unconditional, such that it evaluates that each row is satisfied without regard for any conditional parameters. Here, LENGTHUNIT passes and THERMODYNAMICTEMPERATUREUNIT fails as above, however FORCEUNIT fails now because it is required (not a condition), and does not exist in the file. Exporting applications may use any combination of the allowed units. Importing applications must be able to deal with any combination of the allowed unit. Mixing different unit systems in the same dataset is not allowed. Datasets with mixed unit systems must be rejected.
Checking types	Value-C, Value-A, Type-R, Type-C

Table 17 continued

<table>
<tr>
<td>Concept</td>
<td>
MVC-888 Metric Project Units

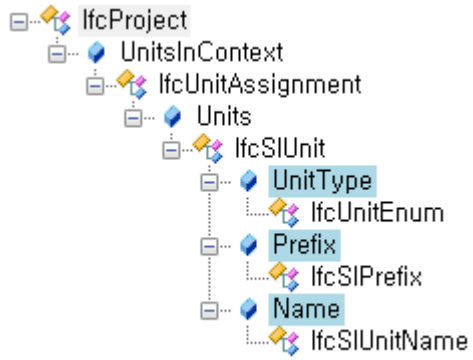
IfcUnitAssignment
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>Units</td><td>Must be list of IfcSiUnit</td></tr>
</table>

IfcSIUnit
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>UnitType</td><td>AreaUnit, LengthUnit, VolumeUnit, MassUnit</td></tr>
<tr><td>Prefix</td><td>Centi, Milli</td></tr>
<tr><td>Name</td><td>Cubic_Metre, Metre, Square_Metre or Gram</td></tr>
</table>

<table>
<tr><th>UnitType</th><th>Prefix</th><th>Name</th></tr>
<tr><td>AREAUNIT</td><td></td><td>SQUARE_METRE</td></tr>
<tr><td>AREAUNIT</td><td>CENTI</td><td>SQUARE_METRE</td></tr>
<tr><td>AREAUNIT</td><td>MILLI</td><td>SQUARE_METRE</td></tr>
<tr><td>LENGTHUNIT</td><td></td><td>METRE</td></tr>
<tr><td>LENGTHUNIT</td><td>CENTI</td><td>METRE</td></tr>
<tr><td>LENGTHUNIT</td><td>MILLI</td><td>METRE</td></tr>
<tr><td>VOLUMEUNIT</td><td></td><td>CUBIC_METRE</td></tr>
<tr><td>VOLUMEUNIT</td><td>CENTI</td><td>CUBIC_METRE</td></tr>
<tr><td>VOLUMEUNIT</td><td>MILLI</td><td>CUBIC_METRE</td></tr>
<tr><td>MASSUNIT</td><td></td><td>GRAM</td></tr>
</table>
</td>
</tr>
<tr>
<td>IfcDoc</td>
<td>


</td>
</tr>
</table>

Table 17 continued

Rule Definition	 UnitType / Prefix / UnitName AREAUNIT / CENTI / SQUARE_METRE AREAUNIT / MILLI / SQUARE_METRE LENGTHUNIT / CENTI / METRE LENGTHUNIT / MILLI / METRE VOLUMEUNIT / CENTI / CUBIC_METRE VOLUMEUNIT / MILLI / CUBIC_METRE MASSUNIT / / GRAM

4.2.8 Geometry representation type

Geometry representation types can be defined in the individual concept templates. An exchange, however, often require diverse representation types such as boundary representation and extrusion. For this condition, the nested concepts (conditional checking for concepts overall) is used by the Move In (or Move Out) feature of IfcDoc to refer concepts using entities defined underneath their root entities. Such nesting works with OR logic such that if at least one nested concept passes, then the outer concept passes. This was a significant change that impacted a lot of areas.

To achieve the described result, a template at the outermost entity containing the relevant attributes such as IfcProductDefinitionShape can be defined, though better to use IfcRoot-based element such as IfcBuildingElement. Then, define rules indicating: (Description=”Brep shape” AND parentcurve=IfcPolyline) OR (Description =” Brep shape” AND (parentcurve=IfcPolyline OR parentcurve=IfcTrimmedCurve). You may also want to look at the IFC2x3 and IFC4 documentation for how geometry is normally constrained – using IfcProductDefinitionShape.Description for this purpose is inconsistent with other usage – IfcRepresentation.RepresentationIdentifier is used to indicate the particular representation (e.g. “Body”, “Surface”, “Axis”, etc.) and IfcText fields are for free-form human-defined text not intended to be used as identifiers.

Table 18 The concept example of geometry representation checking

Requirements	The PCI-088 concept describing Extruded Shape Geometry defines the extruded shape geometry for all rebar. The following diagram says the geometry of the reinforcing is swept disk solid, so the geometry of the rebar in the developed sample model is swept disk solid. A rebar instance can use the representation method of swept disk solid by referring directly to IfcSweptDiskSolid or by referring to IfcMappedItem and follow IfcSweptDiskSolid. Thus, these two ways should be provided so that an instance can be passed if it complies to one of them.
Checking types	Type-S, Type-R, Type-C

Table 18 continued

Concept	PCI-088 Extruded Shape Geometry of Reinforcing Element IfcShapeRepresentation **Attribute** / **Implementation agreements** ContextOfItems: Required RepresentationIdentifier: Label:'Swept disk' RepresentationType: Label: 'CompositeCurve' Items: Required; conceived for possible multiple representations; but usually 3D model. IfcSweptDiskSolid **Attribute** / **Implementation agreements** Directrix: Required; It defines a curve composed of segments that each is defined as a composite curve segment. Radius: Required: positive length measure InnerRadius: Optional: positive length measure StartParameter: Start point on curve, between 0.0 and 1.0 EndParameter: Endpoint on curve, between 0.0 and 1.0 IfcCompositeCurve **Attribute** / **Implementation agreements** Segments: Required; It defines a curve composed of segments that each is defined as a composite curve segment. SelfIntersect: Required

Table 18 continued

IfcDoc

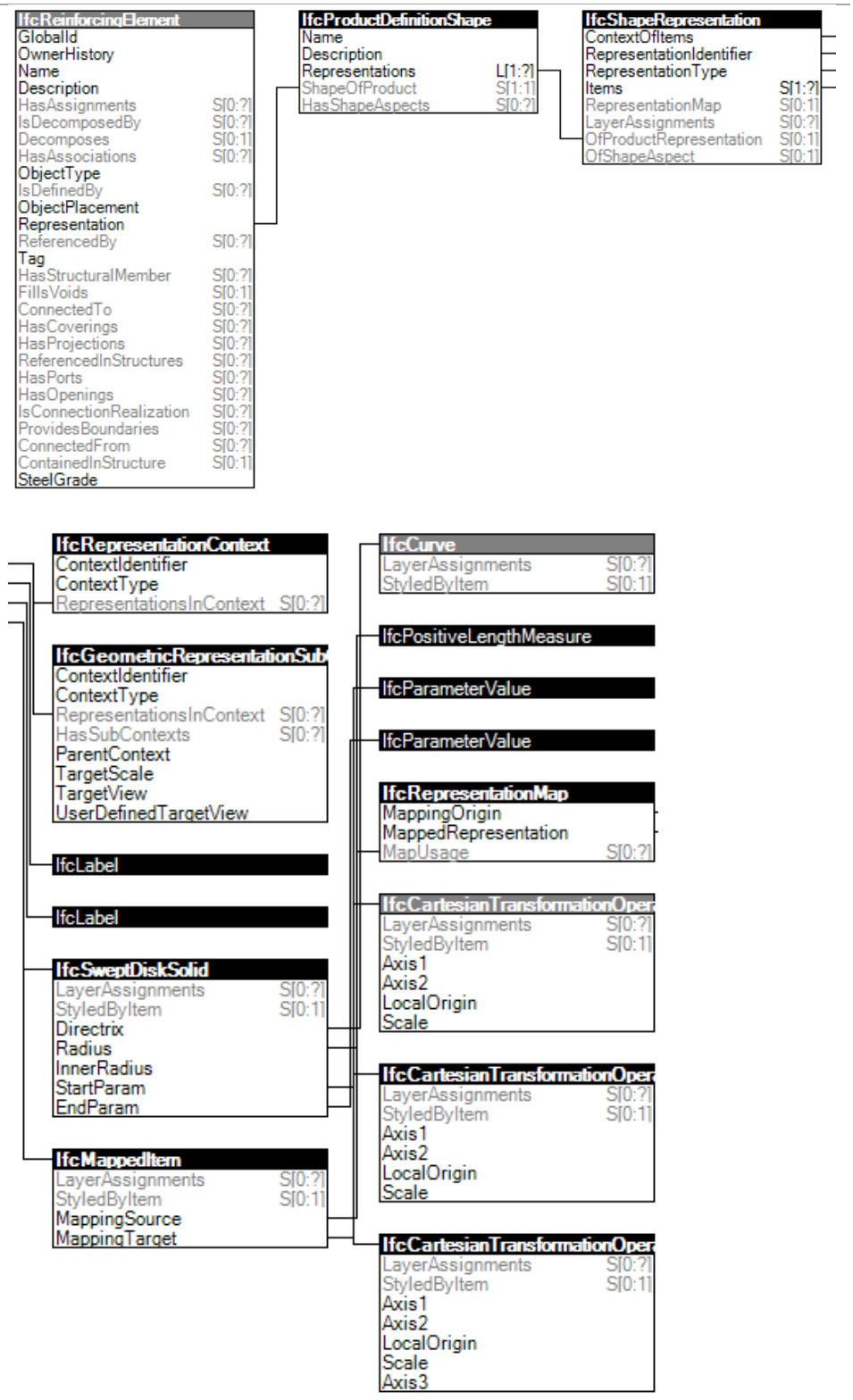


Table 18 continued

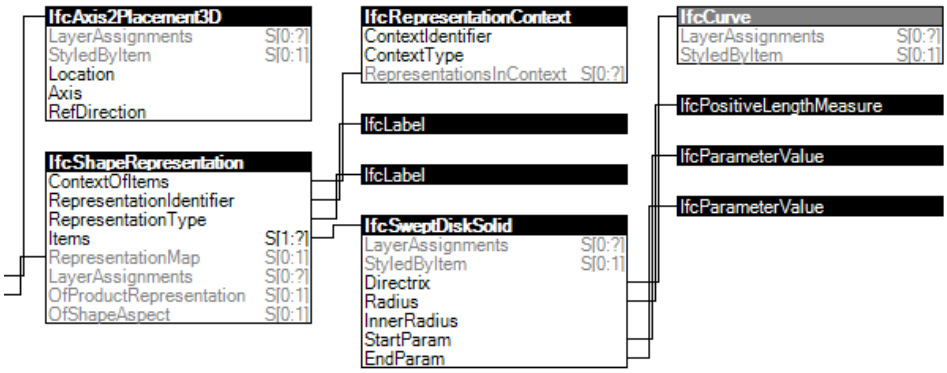
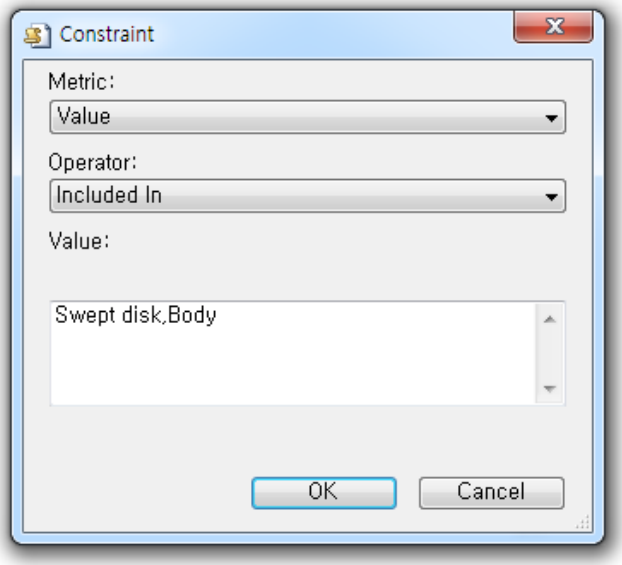
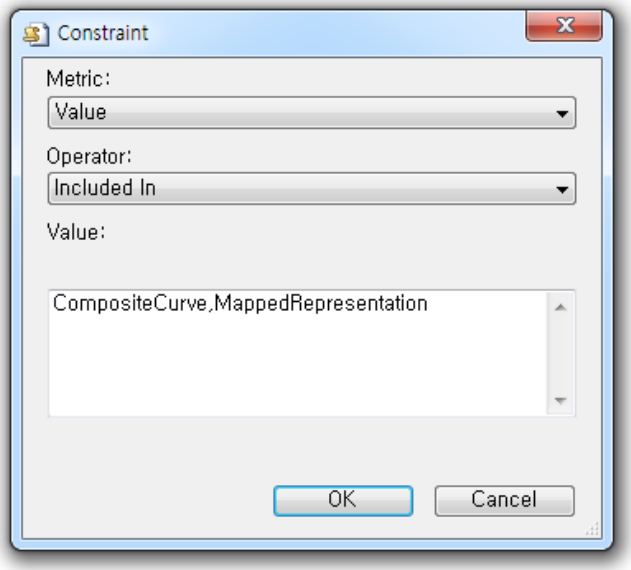
IfcDoc	
Rule Definition	 

Table 19 The example of subtype checking (Lee, Eastman et al. 2015)

<table>
<tr><td>Requirements</td><td>In the PCI-145 concept, the representation type of a precast piece can be defined by Brep or Swept Solid geometric representation types of the IfcProjectionElement entity. Hence, the IfcProjectionElement entity of an IFC instance file can include IfcShapeRepresentation and the Items attribute of IfcShapeRepresentation can refers to one of possible representation types: the subtypes of IfcManifoldSolidBrep are IfcFacetedBrep and IfcFacetedBrepWithVoid, and the subtypes of IfcSweptAreaSolid are IfcExtrudedAreaSolid, IfcRevolvedAreaSolid, and IfcSurfaceCurveSweptAreaSolid. IfcSweptDiskSolid can be used by this concept.</td></tr>
<tr><td>Checking types</td><td>Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-145 Precast Projection Attributes
IfcProjectionElement
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>GUID</td><td>Must be provided</td></tr>
<tr><td>OwnerHistory</td><td>Must be provided, but may contain dummy data</td></tr>
<tr><td>Name</td><td>Optional</td></tr>
<tr><td>Description</td><td>Optional</td></tr>
<tr><td>ObjectType</td><td>Should be “Precast Projection“</td></tr>
<tr><td>ObjectPlacement</td><td>Local placement</td></tr>
<tr><td>Representation</td><td>Geometric representation, Brep or Swept Solid</td></tr>
<tr><td>Tag</td><td>.CORBEL. or similar</td></tr>
</table></td></tr>
</table>

Table 19 continued

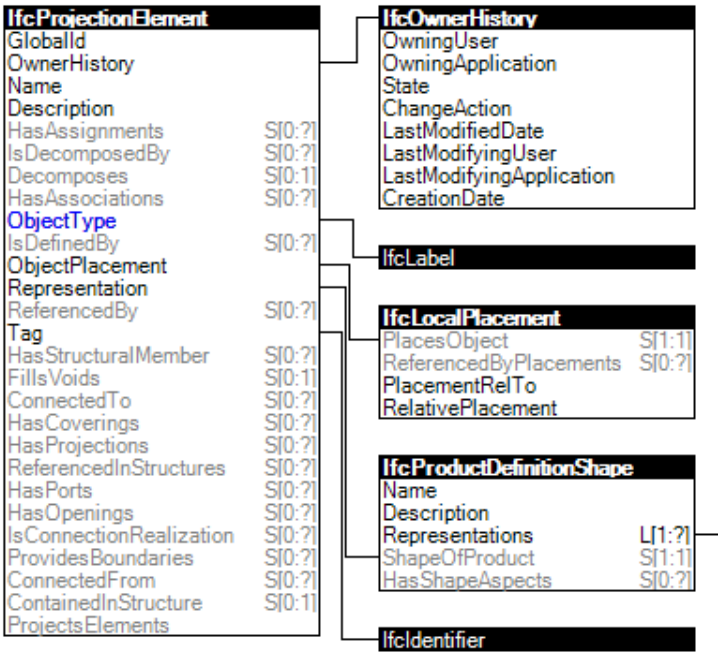
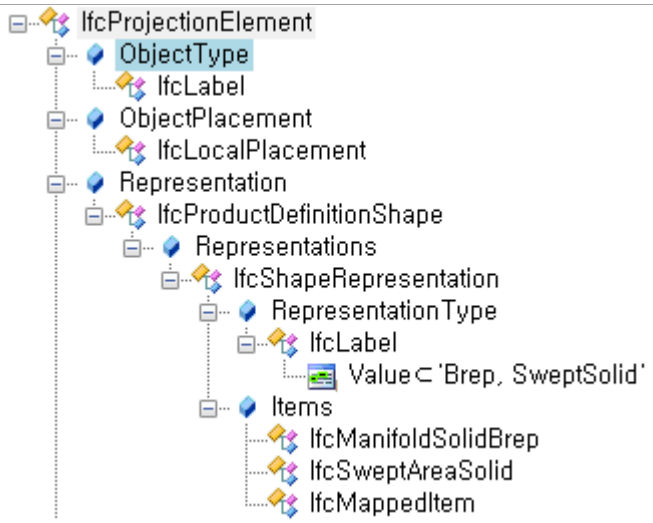
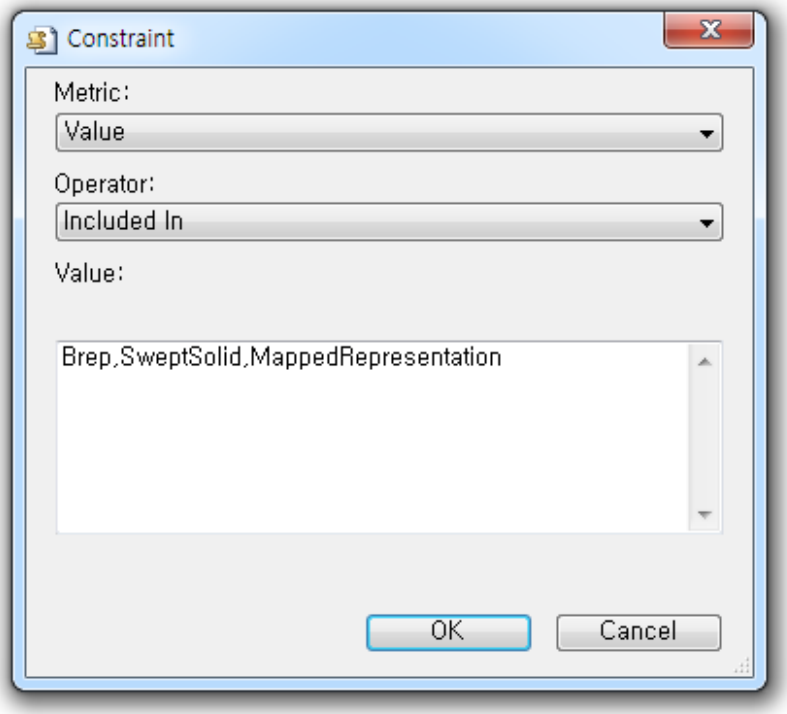
IfcDoc	

Table 19 continued

Rule Definition	 

4.2.9 PropertySet

Defined property sets can be referred by several objects and by to as shared property sets. Thus, property set checking requires a conditional checking logic that can identify the accurate type of a property set. Since numerous properties are related to objects, a name attribute is used to separately apply interpretation of corresponding properties. Objects can embrace property sets through an objectified relationship, IfcRelDefinedByProperties. Each single property values can be defined as required and optional because requirements of such values are determined by types of objects and phases of exchanges. To define extensible properties, parameters for a Name and a HasProperties attributes of a property set are used.

Table 20 Property set

Requirements	PCI-056 (related concepts: PCI-057, 077, 086, 091, 097, 165) Several scenarios can be applied to the property set checking. First, applicable entities in the table below must have an IfcRelDefinesByProperties reference for an IsDefinedBy attribute. IfcRelDefinesByProperties must refers to IfcPropertySet that has 'Pset_PrecastConcreteElementGeneral' as the Property Set name. In addition, HasProperties of IfcPropertySet must refer to IfcPropertySingleValue that contains accurate names and needed NormialValues.
Checking types	Value-C, Type-C, Type-I, Type-D, Type-E

Table 20 continued

Concept

PCI-056 General Properties of Precast Element

IfcPropertySet

Attribute	Implementation agreements
GUID	Must be provided
OwnerHistory	Must be provided, but may contain dummy data
Name	Must be "Pset_PrecastConcreteElementGeneral"
Description	Optional
HasProperties	Must point to the full list of properties defined in the table below; one IfcPropertySingleValue must appear for each property listed in the table
ObjectPlacement	Local placement
Representation	Geometric representation, Brep or Swept Solid
Tag	.CORBEL. or similar

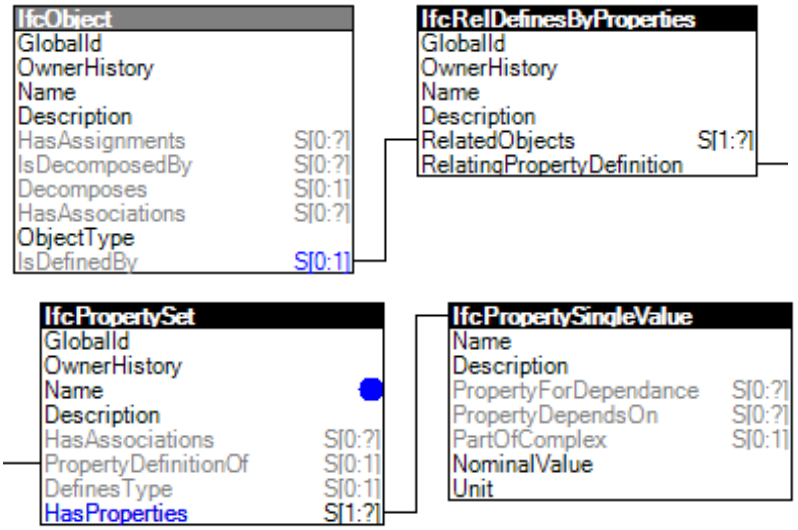
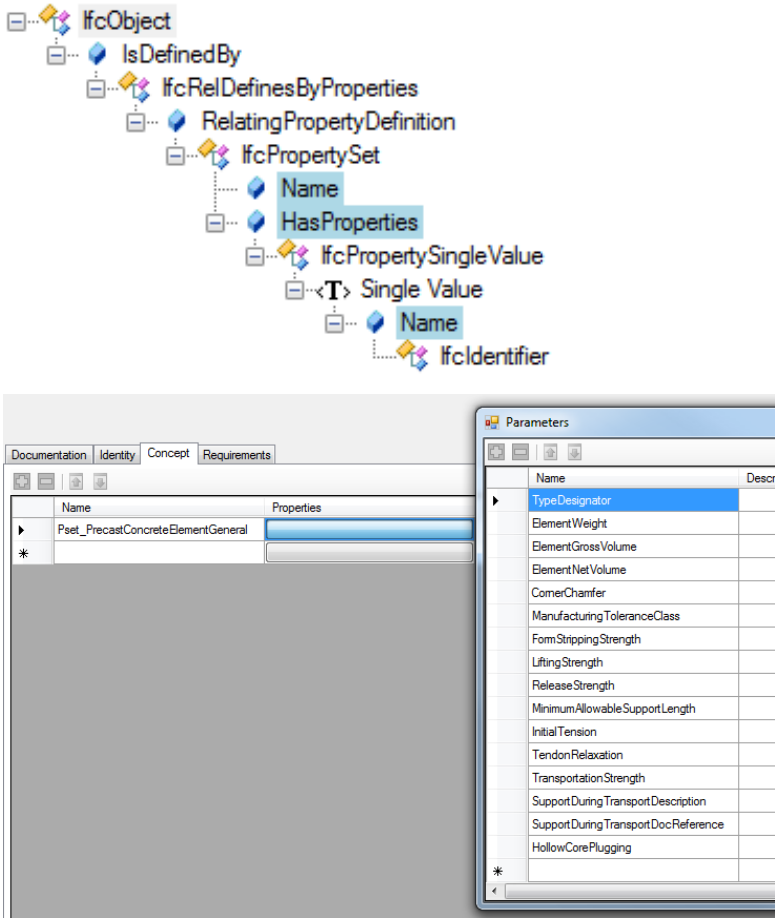
IfcPropertySingleValue - for Pset_PrecastConcreteElementGeneral

Attribute	Implementation agreements
Name	Must be from the table below
Description	Optional
NominalValue	Optional: either $ or from the table below
IfcUnit	Must be from the table below

Table 20 continued

Concept	PropertySet Definition: **PropertySet Name:** Pset_PrecastConcreteElementGeneral **Applicable Entities:** IfcBeam IfcBeamType IfcBuildingElementPart IfcBuildingElementPartType IfcBuildingElementProxy IfcBuildingElementProxyType IfcColumn IfcColumnType IfcCovering IfcCoveringType IfcCurtainWall IfcCurtainWallType IfcFooting IfcFootingType IfcMember IfcMemberType IfcPile IfcPileType IfcRailing IfcRailingType IfcRamp IfcRampType IfcRampFlight IfcRampFlightType IfcRoof IfcRoofType IfcSlab IfcSlabType IfcStair IfcStairType IfcStairFlight IfcStairFlightType IfcWall IfcWallType IfcWallStandardCase **Applicable Type Value:** **Definition:** Definition from IAI: Production and manufacturing related properties common to different types of precast concrete elements. The Pset can be used by a number of subtypes of IfcBuildingElement. If the precast concrete element is a sandwich wall panel each structural layer or shell represented by an IfcBuildingElementPart may be attached to a separate Pset of this type, if needed. Some of the properties apply only for specific types of precast concrete elements.

Table 20 continued

IfcDoc	
Rule Definition	

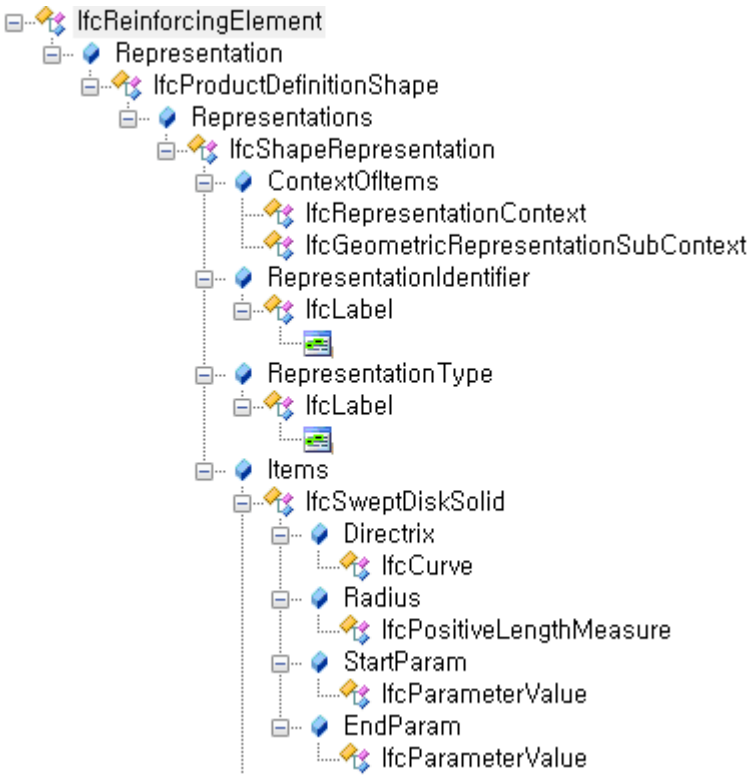
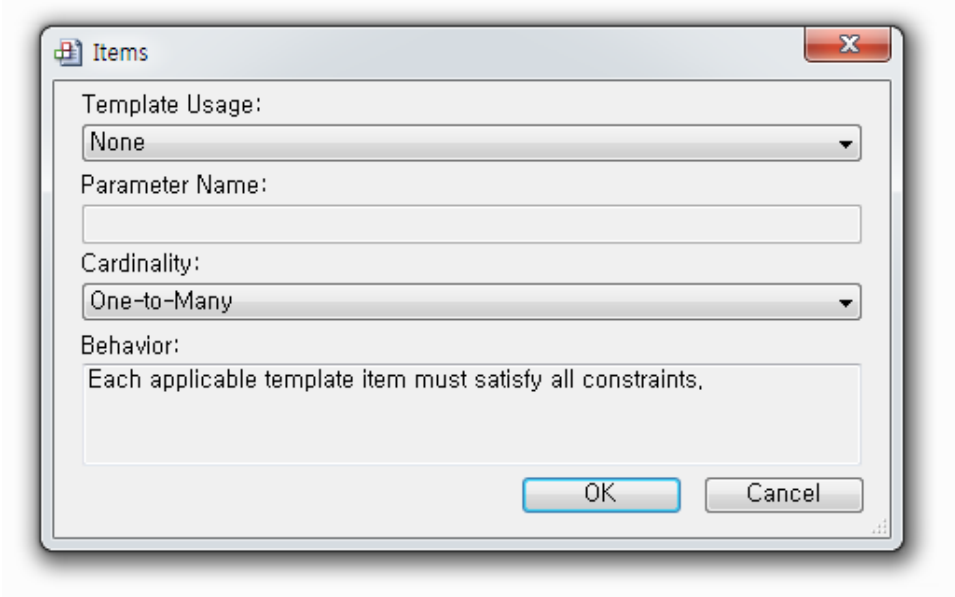
4.2.10 Mandatory and optional

Three layers are applied to demonstrate a mandatory and optional setting. The first layer is to restrict the number of values and references through a cardinality setting that supports defining optional and mandatory for specific attribute. In case of that the lower bound is zero, this attribute is optional. In other words, if the upper bound is greater than one, this attribute is mandatory. Manipulating the required number of an attribute can manage the optional and mandatory setting, which is critical to validate instances. In particular, all of the inverse relationship has [0:?] cardinality setting. Thus, users can manipulate this lower and upper bound setting for limiting the number of references.

Table 21 Mandatory and optional – first layer

<table>
<tr><td>Requirements</td><td>The PCI-088 concept defines ContextOfItems and Items required. Thus, the cardinalities of the ConextOfItems and the Items attributes should be greater than 1.</td></tr>
<tr><td>Checking types</td><td>Type-A, Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-088 Extruded Shape Geometry of Reinforcing Element
IfcShapeRepresentation
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>ContextOfItems</td><td>Required</td></tr>
<tr><td>RepresentationIdentifier</td><td>Label:’Swept disk’</td></tr>
<tr><td>RepresentationType</td><td>Label: ‘CompositeCurve’</td></tr>
<tr><td>Items</td><td>Required; conceived for possible multiple representations; but usually 3D model.</td></tr>
</table></td></tr>
</table>

Table 21 continued

IfcDoc	
Rule Definition	

The second layer is about setting mandatory and optional for each parameter. In a template level, parameters can be defined so that users define diverse requirements regarding entity types. Thus, possible options for an attribute should be defined in a parameter under an assigned entity. Sometimes each parameter should be defined mandatory and optional. A mandatory parameter must be met by an instance file and an optional one.

Table 22 Mandatory and optional – second layer

<table>
<tr><td>Requirements</td><td>The PCI-103 uses parameters for Tag and PredefinedType so that entities referring this concept can define diverse values for both attributes. Mandatory and optional setting can be set to each row of parameters.</td></tr>
<tr><td>Checking types</td><td>Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-103 Aggregation of Reinforcing Assemblies
IfcElementAssembly
<table><tr><th>Attribute</th><th>Implementation agreements</th></tr><tr><td>GlobalId</td><td>Must be provided</td></tr><tr><td>OwnerHistory</td><td>Must be provided</td></tr><tr><td>Name</td><td>Optional</td></tr><tr><td>Description</td><td>Optional</td></tr><tr><td>Tag</td><td>Required; Aggregates reinforcing assemblies: if mesh aggregation, value is “MESH ASSEMBLY”, if rebar aggregation, value is ‘REBAR ASSEMBLY’, if tendon aggregation, value is ‘TENDON ASSEMBLY’</td></tr></table></td></tr>
</table>

Table 22 continued

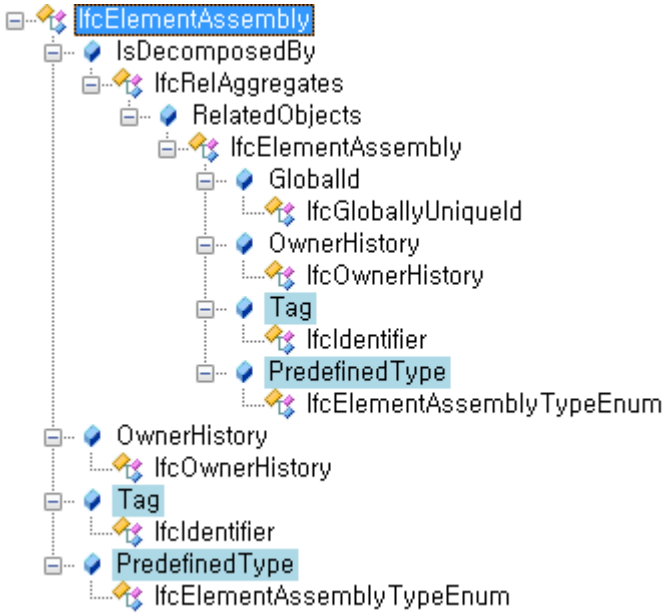
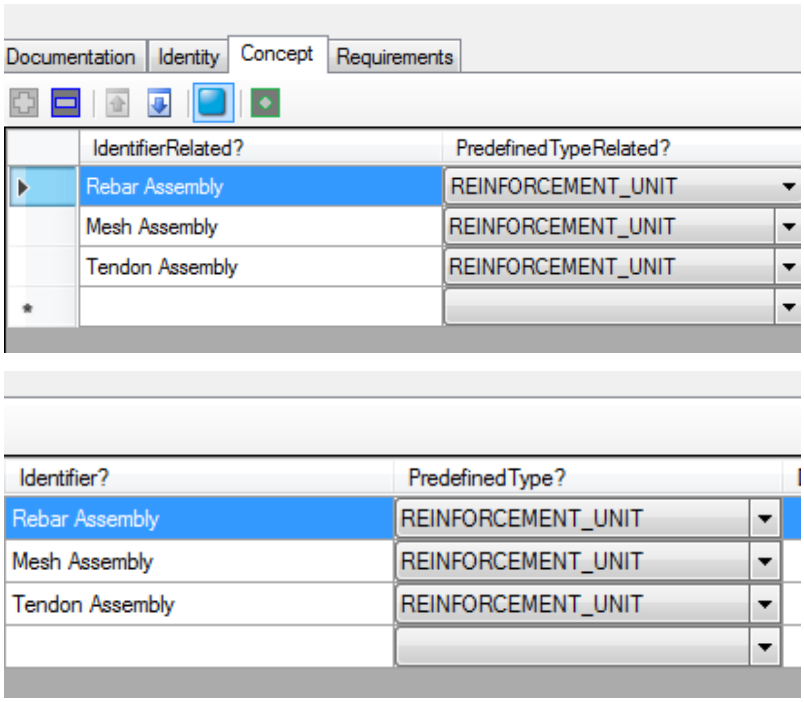
<table>
<tr><td>Concept</td><td><table>
<tr><td>Decomposes</td><td>Decomposes an assembly to its reinforcement element aggregation constituents, which is usually a rebar assembly.</td></tr>
<tr><td>PredefinedType</td><td>Required; All are aggregations, not groups</td></tr>
</table></td></tr>
<tr><td>IfcDoc</td><td> 
</td></tr>
</table>

Table 22 continued

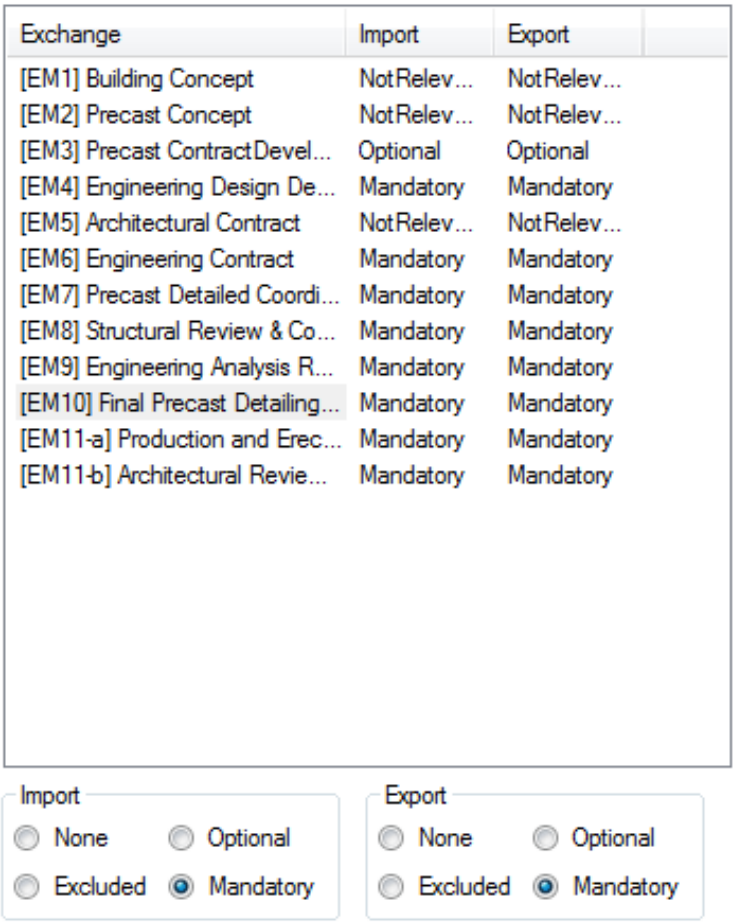
Rule Definition	

The third layer is to define mandatory/optional requirements of at a concept level according to exchanges. The MVD has a set of exchange models and each exchange model consists of concepts. Since each concept is employed by several entities, assigned concepts must be indicated pertaining to what an exchange use an associated concept. As shown in the below picture, this application offers four options available: None, Optional, Excluded, and Mandatory. Such choices can be determined based on exchanges as well as on an import/export setting.

Table 23 Mandatory and optional – third layer

<table>
<tr><td>Requirements</td><td>The PCI-088 is used by twelve EMs by setting None, Optional, Excluded and Required for import and export features. Based on these setting, assigned concepts are executed and used for validation.</td></tr>
<tr><td>Checking types</td><td>Type-S, Type-R</td></tr>
<tr><td>Concept</td><td>PCI-088 Extruded Shape Geometry of Reinforcing Element
IfcShapeRepresentation
<table>
<tr><th>Attribute</th><th>Implementation agreements</th></tr>
<tr><td>ContextOfItems</td><td>Required</td></tr>
<tr><td>RepresentationIdentifier</td><td>Label:’Swept disk’</td></tr>
<tr><td>RepresentationType</td><td>Label: ‘CompositeCurve’</td></tr>
<tr><td>Items</td><td>Required; conceived for possible multiple representations; but usually 3D model.</td></tr>
</table></td></tr>
</table>

Table 23 continued

Rule Definition	Exchange Requirements: 

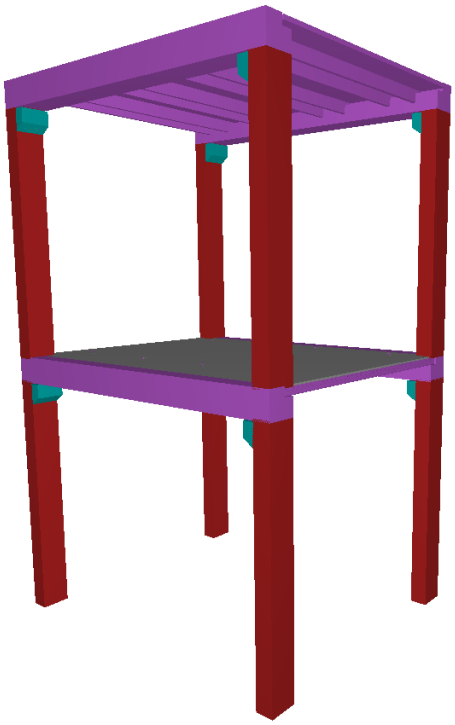
CHAPTER 5 EVALUATIONS AND CONTRIBUTIONS

5.1 Validation reports and evaluation

The IfcDoc tool available from the bSI website as a windows binary file and an open source is one of applications that can implement rule logic. Several types of rule-checking features for executing rule logic were developed and embedded on the IfcDoc tool by Constructivity™. Such execution structures operate on the diverse entities, attributes, relationships, and data types in IFC. To evaluate rule logic and associated checking features, the author uses new checking features to define the IfcDoc file for PCI MVD. An IfcDoc file, which is generated based on the version of IFC to be represented such as IFC Release 2x3 or Release 4 baselines, consists of concept definitions and rule sets. This IfcDoc file has PCI concepts that include corresponding rules. In addition, within the running of IfcDoc, a user can define rule checks as a set based on the concept with which the rules are associated. Rules of a set of concepts are organized and applied to each EM. The IfcDoc file of the PCI MVD supports twelve export EMs with automatic rule checking. Through this automated and advanced checking platform, precast sample models were tested and reviewed by the validation process to ensure the interoperability of the data exchange of BIM data models.

This section shows how the self-validation tool and process work. Figure 21 illustrates a precast concrete sample model that includes beams, columns, slabs, toppings, reinforcing bars, meshes, and tendons. The sample model, which is supposed to be used for EM 10 validation, consists of several components and assemblies, shown in Figure 22: beam

details, including a rebar into stirrups into a beam and a lifting anchor in beams, assembly details about a slab with connection joints, a slab assembly with reinforcing bars, and corbel components. Figure 22 represents the primary parts of components and assemblies in a transparent view, which clearly illustrates how the components and assemblies are organized and related.



Object Type	Number
Beam	13
Column	8
Topping (BUILDINGELEMENTPART)	2
Assembly	21
Rebar	50
Mesh	20
Tendon	8

Figure 21 Test model and its core elements for testing EM 10

To evaluate an IFC instance file according to a specific EM that users need, the IfcDoc tool imports an IfcDoc file with the objective of facilitating interactive debugging of instance models.

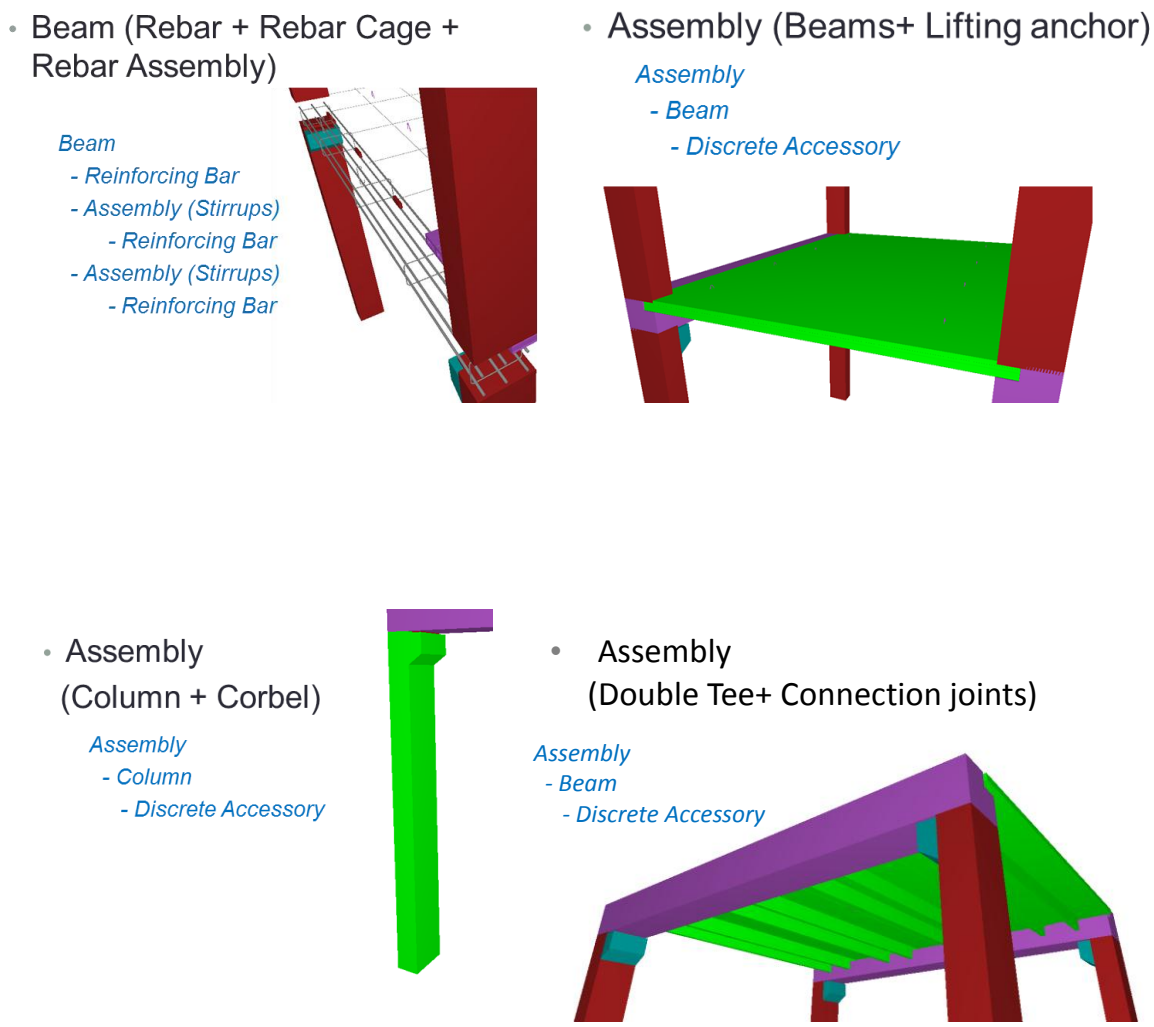


Figure 22 Details of components and assemblies of the precast sample model (Lee and Eastman 2015)

This application generates two validation reports represented in a user interface and an HTML format. Once a P21 file is validated, these two validation outputs are generated at the same time. Figure 23 shows the UI report that has a main window with a separate pane. The main window contains a relational diagram and constraint requirements and the separate pane represents particular object instances for an efficient debugging process. Using a color-coding method, the UI validation report represents the locations of checking errors. The entities, attributes, and relations passed in validation are represented as green and any invalid ones are illustrated as red.

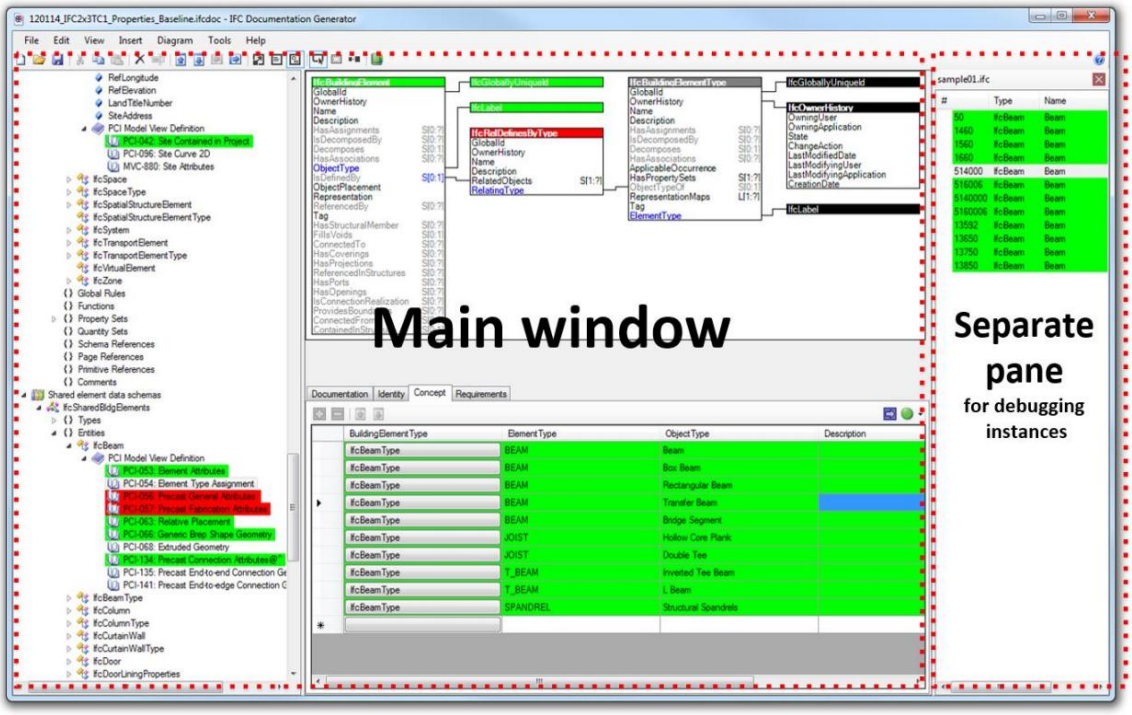


Figure 23 A visual validation report of the IfcDoc tool (Lee and Eastman 2015)

Figure 24 represents a color-coded report that illustrates validation outputs for an instance a user selected. A user can choose one concept in the left pane for a concept structure and select one instance in the separate pane on the right. Then, the IfcDoc represents specific

validation results for the selected instance using a highlighted diagram in the main window. Since the report consists of color-coded entities and attributes, a user can intuitively find the causes of errors pertaining to validation for specific exchange requirements. In particular, a visually demonstrated output in a constraint tab can provide information for explicit outputs of optional checking for each constraint rule.

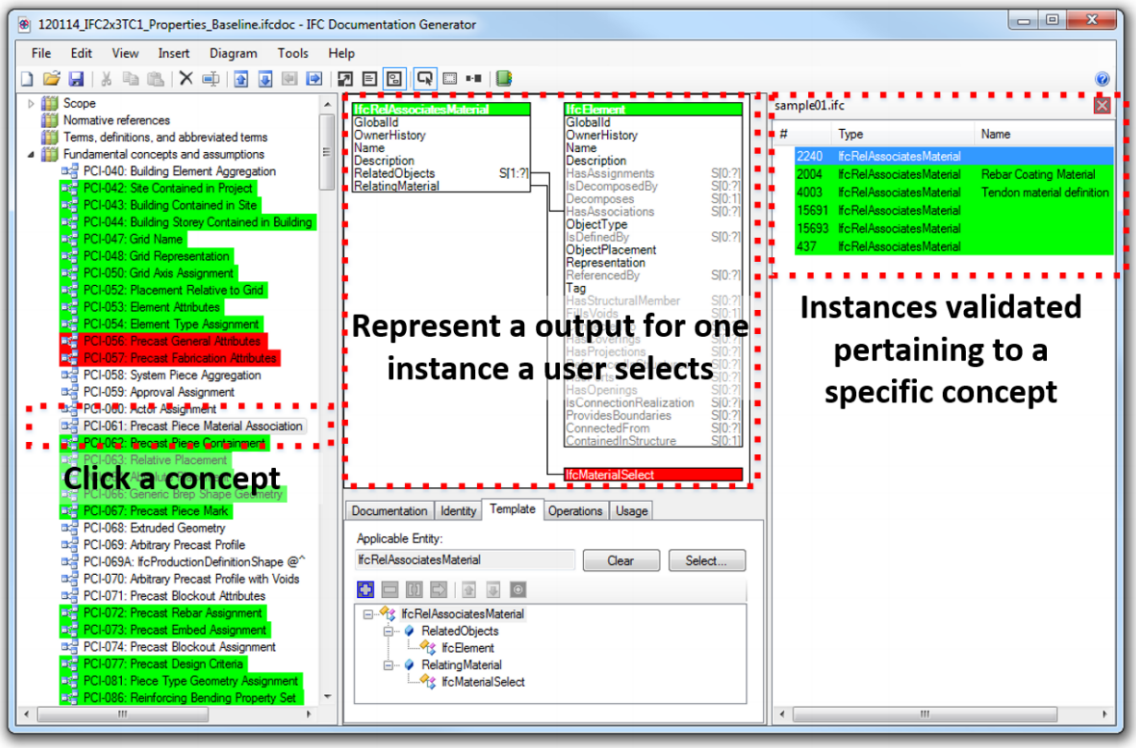


Figure 24 A color-coded report for one instance (Lee and Eastman 2015)

Figure 25 shows validation of the IfcIdentifier attribute of IfcElementAssembly. Because the value for IfcIdentifier can be Rebar Assembly, Mesh Assembly, Tendon Assembly, or Custom Assembly, one constraint expression for Rebar Assembly that a P21 file satisfies is flagged in green. With regard to interactive validation, this visual report also enables changes to be made to rules such that a user can see the impact on a file immediately

without re-running tests. In other words, once a user revises an instance, the report is immediately updated to reflect the update according to relevant checking.

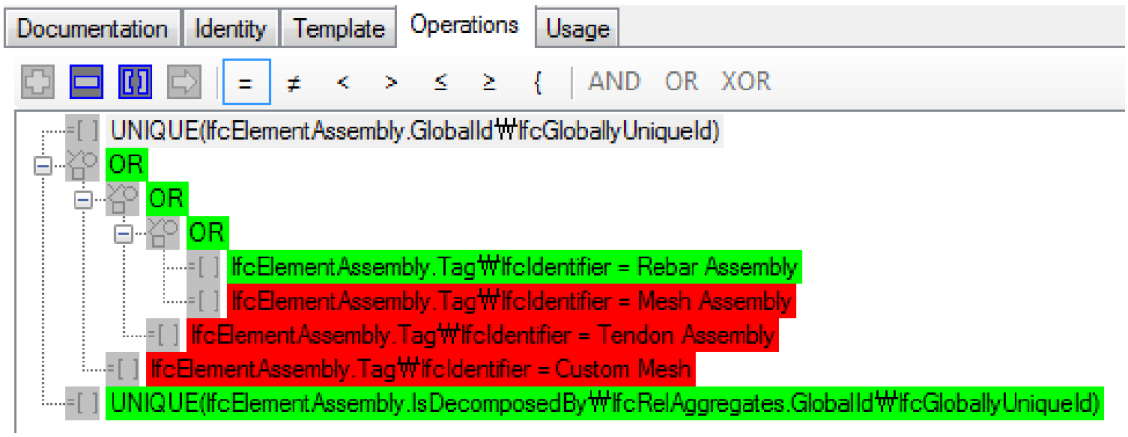


Figure 25 Explicit results of optional checking

The UI report shows associated concepts and represents which IFC instance has errors color coded in red. As shown in Figure 26, if the user clicks on an erroneous concept, IfcDoc lists instances applied to the entity (right column) and the entity structure of the concept (middle column). Green indicates concepts that were passed in the tests; and those with no color were not used in validation.

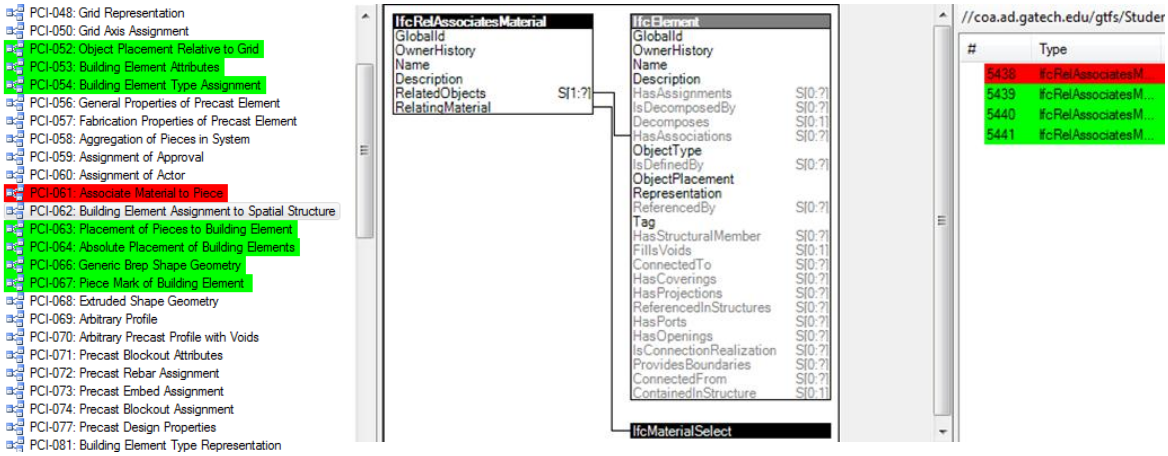


Figure 26 UI report showing errors in the PCI-061 concept and #5438 instance

The other debug report is in the HTML format, which can be stored and printed as a file. The HTML error report can be easily shared by a printed document and an email. The concept evaluates and figure out the path structure of the RelatedObjects attribute of IfcElement. It checks elements that are being assigned material that allows instances to have types. The project IFC file instances are shown in the left column, whose pathway is in the STRUCTURE column, and the constraint rule is assessed in the right column. In the table underneath each concept such as PCI-061: Associate Material to Piece in Figure 27, the plus (+) sign indicates pass, and other notes indicate fail. Figure 27 represents an HTML validation report that illustrates the FAIL results of PCI-061 validation (associated material with an element). The FAIL was the result of incorrect references: RelatedObjects of the IFCRELASSOCIATESMATERIAL entity must be only IfcDefinitionSelect. Figure 28 shows that the instance encompasses IFCGRIDAXIS, which causes the error because IfcGridAxis was assigned to the list receiving the material, which was clearly an error.

Validation Results

Instance File	C:\YongCheol\Project, Work\IfcDoc Extension\Sample files\garage_syms_4.ifc
Project File	C:\YongCheol\Dropbox\Dropbox\PCI-NBIMS\ifcDOC\051815_IFC2x3TC1_Properties_Baseline.ifcdoc
Model View	PCI Model View Definition
Exchange	[EM1] Building Concept
Tests Executed	52
Tests Passed	30
Tests Ignored	0
Tests Percentage	57%

IfcSlab (31)

▶ PCI-040: Building Element Aggregation
▶ PCI-053: Building Element Attributes - [FAIL]
▶ PCI-054: Building Element Type Assignment
▶ PCI-063: Placement of Pieces to Building Element
▶ PCI-068: Extruded Shape Geometry - [FAIL]

IfcSite (1)

▶ PCI-042: Site Contained in Project
▶ PCI-096: Site Geometric Curve Representation - [FAIL]
▶ MVC-880: Site Attributes - [FAIL]

IfcBuilding (1)

▶ PCI-043: Building Contained in Site
▶ MVC-893: Building Attribute - [FAIL]

IfcRelAssociatesMaterial (53)

▼ PCI-061: Associate Material to Piece - [FAIL]

Instance	Structure	Constraints
#5438	.RelatedObjects\IfcElement	+
#5439	+	+
#5440	+	+
#5441	+	+

Figure 27 HTML validation report and FAIL in a material property set

#5438=
IFCRELASSOCIATESMATERIAL('3taLHh_IKHv9iZz6q4uvbE',#11,$,$,(#128,#254,#380,#530,#854,#1178,#1502,#3659,#3662,#3665,#3668,#3671,#3674,#3677,#3680,#3683,#3686,#3689,#3692,#3695,#3698,#3701,#3704,#3707,#3710,#4029,#4258,#4582,#4906,#5230),#5406);

#3659= IFCGRIDAXIS('UY-1',#1120,.T.);
#3662= IFCGRIDAXIS('UY-2',#1130,.T.);
#3665= IFCGRIDAXIS('UY-3',#1140,.T.);

Figure 28 Instance of a sample model

To identify errors and their locations within instances, users can use two UI and HTML validation reports simultaneously. As an example, Figure 29 shows another validation check on the test file that deals with the IFC spatial structure. IFC organizes building models within a Project > Site > Building > Storey Spatial hierarchy. An error of this example was found in the test model in PCI-062 and in entity #5456. The elements assigned to the IFC spatial structure to Building is erroneous. Users can keep track of the error using both validation reports significantly helpful for users.

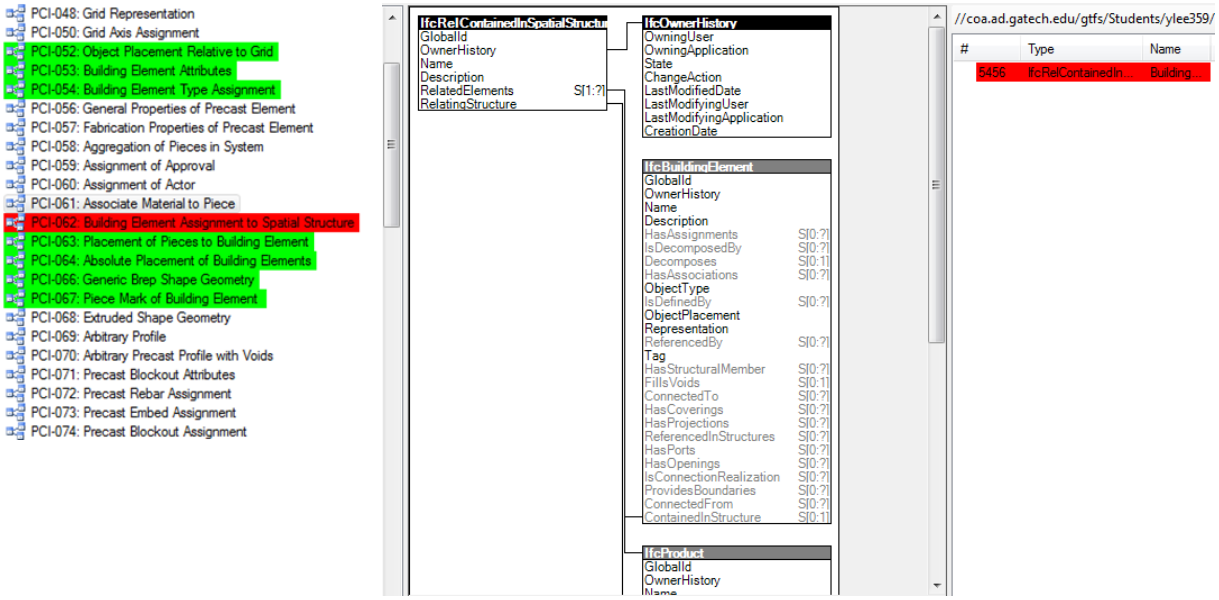


Figure 29 PCI-062 validation reports- the IFC spatial structure to Building

5.2 Implementation of rule logic for checking instances

To evaluate rule logic and this rule-checking process, this section shows examples of implementation and cases of validation using diverse IFC instance files. Such case studies provide large numbers of error reports and the conditions they identify. Using the twelve subset exchange requirements using 96 Concepts, this validation process represents IFC debugging report methods. The evaluation and implementation process consists of diverse validation of IFC instance files exported from diverse software companies using the PCI MVD, IFC Coordination View 2.0, and CObie. Assessment of MVDs can be executed by comparing the new model structures and concepts that are refined and revised by rule-based concept definitions of the precast concrete MVD and its use in documenting and validating this structure.

The objective of assessment is to confirm the usability, consistency, and applicability of rule logic regarding other domains and to promote the use of IfcDoc for validation within the overall development of the MVD. In addition, this evaluation process is expected to reveal its benefits and execution so that software companies can debug import and export systems using the rule-based validation process within the constraints of availability and applicability. The criteria consist of (1) identifying the number of implementable rules, (2) evaluating the percentage of the reuse of existing concept descriptions, (3) comparing the time and effort expended on defining concepts and MVDs, and (4) assessing the efficiency of validating and documenting concepts and MVDs.

5.2.1 Rule definitions for diverse model views

This dissertation entailed the identification and development of rule logic and checking features based on the requirements of the PCI MVD to identify the extent to which such rule logic also covers the rule definitions of other model views. In order to assess the research hypothesis, the rule logic and validation structures of the PCI model view are verified by applying them to other model views. In this section, the IfcDoc files of IFC Coordination View 2.0 and COBie were edited to embed rule sets.

(1) IFC Coordination View 2.0 (CV)

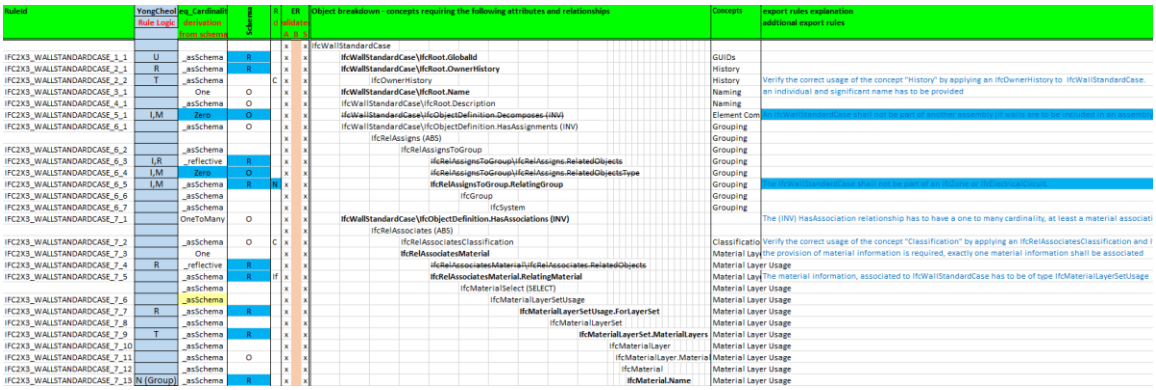
The bSI website states that the Coordination View, which has been widely used in AEC industries, is the first model view developed by bSI (buildingSMARTInternational 2015). The primary objective of the CV is to define standardized specifications for BIM data exchange between architectural, structural engineering, and mechanical industries during the design phase (buildingSMARTInternational 2015). To support coordination of design data among diverse disciplines, the CV includes definitions of architectural, structural, building service, project spatial structure system elements. These specifications are defined in a concept format: The total number of concepts for applicable entities and relationships is 49(buildingSMARTInternational 2015). Concept descriptions define specifications of a name, an aggregation, a material, a shape representation, a connection, an assignment, a property set, and others. To efficiently represent these specifications, the author defined modularized concept templates on IfcDoc V9.4 so that they can describe 49 concept descriptions as shown in Table 24. In addition, corresponding rules attached to each entity were added into the IfcDoc file.

Table 24 Root concepts of IFC CV 2.0 (buildingSMARTInternational 2015)

Architectural and structural elements (shell and core structure)
Wall Standard Case, Wall, Beam, Column, Slab, Stair, Ramp, Railing, Curtain Wall, Roof, Building Element Proxy
Building service elements (predominately)
Distribution Element, Distribution Flow Element, Distribution Control Element, Energy Conversion Device, Flow Controller, Flow Fitting, Flow Moving Device Flow Segment, Flow Storage Device, Flow Terminal, Flow Treatment Device Transport Element, Distribution Chamber Element
Architectural elements (predominately)
Window, Door, Covering, Furnishing Element
Structural elements (predominately)
Member, Plate, Footing, Pile, Fastener, Mechanical Fastener, Discrete Accessory, Element Assembly, Reinforcing Bar, Reinforcing Mesh, Tendon, Tendon Anchor
Project, spatial structure and system elements
Project, Site, Building, Building Storey, Space, Group, System, Zone, Grid

The purpose of this CV definition is to see whether rule logic identified from PCI can execute specifications for other types of model views. Thus, as shown in Figure 30, which indicates requirements of IfcWallStandardCase, the author put rule logic in the second column associated with rules of attributes. The complete IfcDoc CV V2.0 file

defined by the most recent version of IfcDoc will be greatly beneficial and helpful to software companies who use this model view validation tool to develop their IFC interface. The bSI website shows several software vendors who got CV2.0 certification from bSI such as Autodesk, Tekla, and Bentley Systems[1].



RuleId	YongCheol Rule Logic	eq_Cardinality mirrowing from schema	Schema	R	Eft validate A B C	Object breakdown - concepts requiring the following attributes and relationships	Concepts	export rules explanation additional export rules
				x	x	IfcWallStandardCase		
IFC2X3_WALLSTANDARDCASE_1_1	U	_asSchema	R	x	x	**IfcWallStandardCase\IfcRoot.GlobalId**	GUIDs	
IFC2X3_WALLSTANDARDCASE_2_1	R	_asSchema	R	x	x	**IfcWallStandardCase\IfcRoot.OwnerHistory**	History	
IFC2X3_WALLSTANDARDCASE_2_2	T	_asSchema	C	x	x	IfcOwnerHistory	History	Verify the correct usage of the concept "History" by applying an IfcOwnerHistory to IfcWallStandardCase.
IFC2X3_WALLSTANDARDCASE_3_1		One	O	x	x	**IfcWallStandardCase\IfcRoot.Name**	Naming	an individual and significant name has to be provided
IFC2X3_WALLSTANDARDCASE_4_1		_asSchema	O	x	x	IfcWallStandardCase\IfcRoot.Description	Naming	
IFC2X3_WALLSTANDARDCASE_5_1	I,M	Zero	O	x	x	~~IfcWallStandardCase\IfcObjectDefinition.Decomposes (INV)~~	Element Com	An IfcWallStandardCase shall not be part of another assembly (if walls are to be included in an assembly
IFC2X3_WALLSTANDARDCASE_6_1		_asSchema	O	x	x	IfcWallStandardCase\IfcObjectDefinition.HasAssignments (INV)	Grouping	
				x	x	IfcRelAssigns (ABS)	Grouping	
IFC2X3_WALLSTANDARDCASE_6_2		_asSchema		x	x	IfcRelAssignsToGroup	Grouping	
IFC2X3_WALLSTANDARDCASE_6_3	I,R	_reflective	R	x	x	~~IfcRelAssignsToGroup\IfcRelAssigns.RelatedObjects~~	Grouping	
IFC2X3_WALLSTANDARDCASE_6_4	I,M	Zero	O	x	x	~~IfcRelAssignsToGroup\IfcRelAssigns.RelatedObjectsType~~	Grouping	
IFC2X3_WALLSTANDARDCASE_6_5	I,M	_asSchema	R	N	x	x	**IfcRelAssignsToGroup.RelatingGroup**	Grouping	The IfcWallStandardCase shall not be part of an IfcZone or IfcElectricalCircuit.
IFC2X3_WALLSTANDARDCASE_6_6		_asSchema		x	x	IfcGroup	Grouping	
IFC2X3_WALLSTANDARDCASE_6_7		_asSchema		x	x	IfcSystem	Grouping	
IFC2X3_WALLSTANDARDCASE_7_1		OneToMany	O	x	x	**IfcWallStandardCase\IfcObjectDefinition.HasAssociations (INV)**		The (INV) HasAssociation relationship has to have a one to many cardinality, at least a material associati
				x	x	IfcRelAssociates (ABS)		
IFC2X3_WALLSTANDARDCASE_7_2		_asSchema	O	C	x	x	IfcRelAssociatesClassification	Classificatio	Verify the correct usage of the concept "Classification" by applying an IfcRelAssociatesClassification and I
IFC2X3_WALLSTANDARDCASE_7_3		One			x	x	**IfcRelAssociatesMaterial**	Material Lay	the provision of material information is required, exactly one material information shall be associated
IFC2X3_WALLSTANDARDCASE_7_4	R	_reflective	R		x	x	~~IfcRelAssociatesMaterial\IfcRelAssociates.RelatedObjects~~	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_5		_asSchema	R	rf	x	x	**IfcRelAssociatesMaterial.RelatingMaterial**	Material Lay	The material information, associated to IfcWallStandardCase has to be of type IfcMaterialLayerSetUsage
		_asSchema			x	x	IfcMaterialSelect (SELECT)	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_6		_asSchema			x	x	IfcMaterialLayerSetUsage	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_7	R	_asSchema	R		x	x	**IfcMaterialLayerSetUsage.ForLayerSet**	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_8		_asSchema			x	x	IfcMaterialLayerSet	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_9	T	_asSchema	R		x	x	**IfcMaterialLayerSet.MaterialLayers**	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_10		_asSchema			x	x	IfcMaterialLayer	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_11		_asSchema	O		x	x	IfcMaterialLayer.Material	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_12		_asSchema			x	x	IfcMaterial	Material Layer Usage	
IFC2X3_WALLSTANDARDCASE_7_13	N (Group)	_asSchema	R		x	x	**IfcMaterial.Name**	Material Layer Usage	

Figure 30 Specifications of IfcWallStandardCase

[1] buildingSmart International web site for software companies that received IFC CV2.0 certification. (http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc2x3-cv-v2.0-certification/participants)

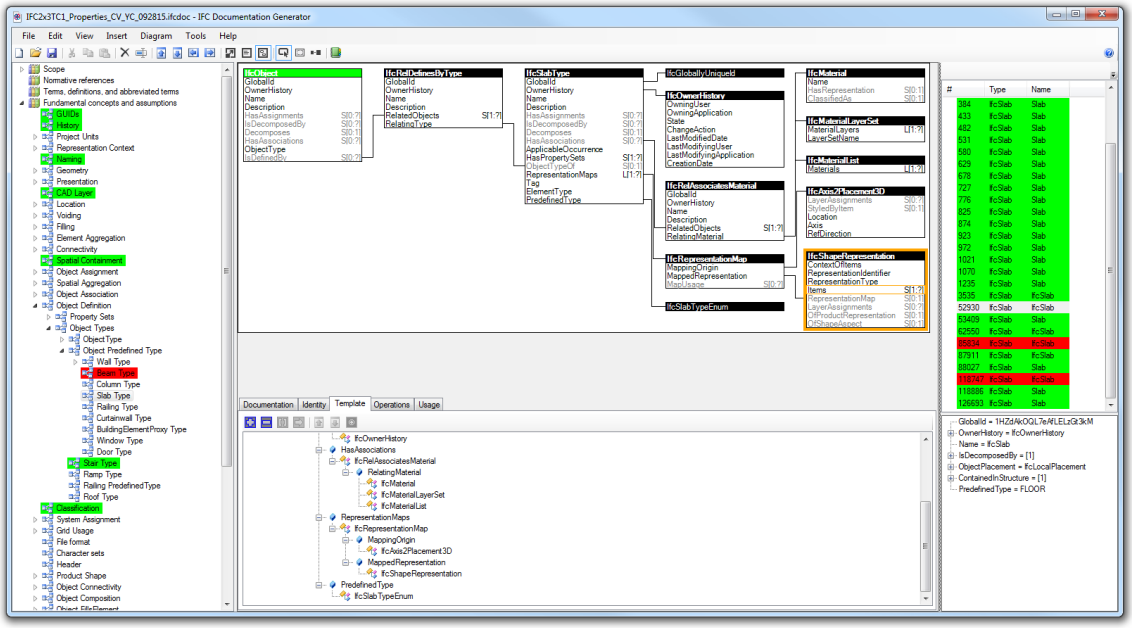
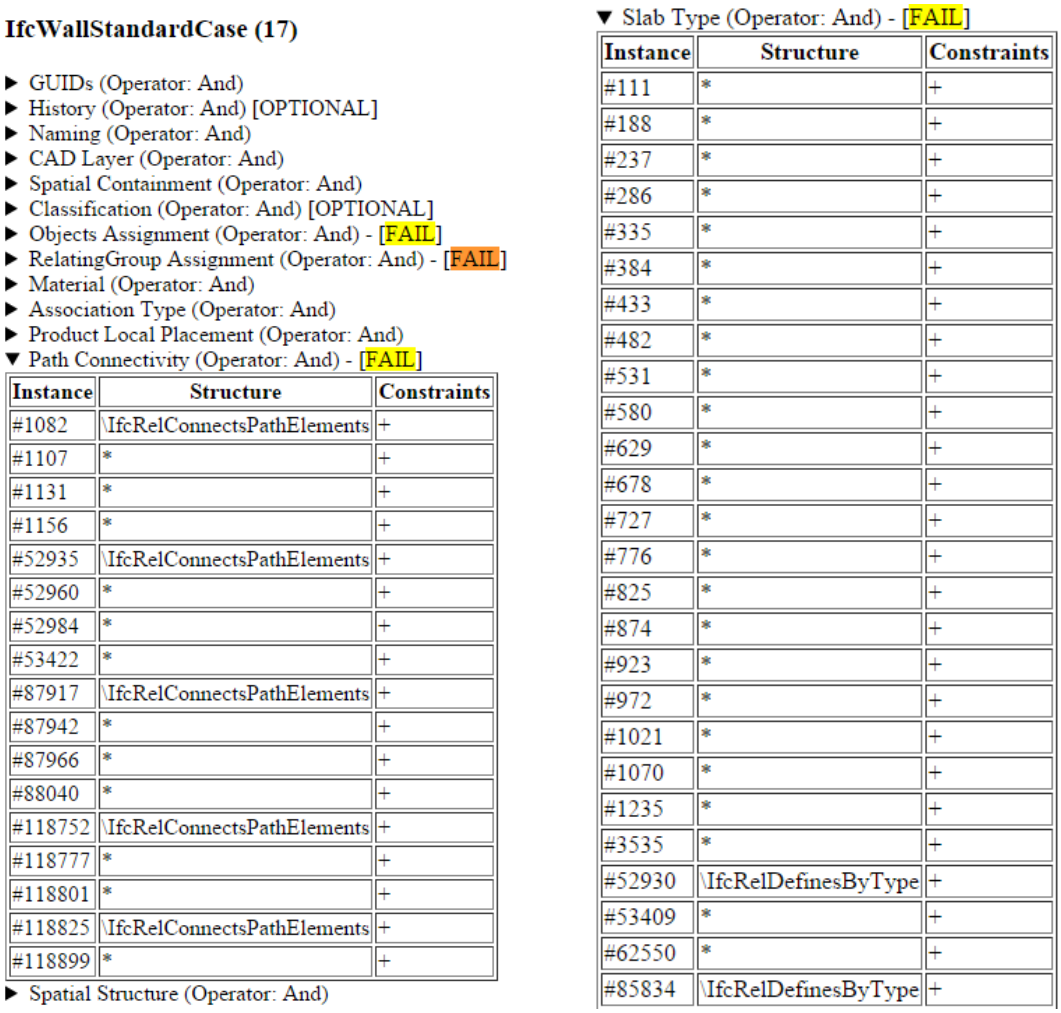


Figure 31 Validation report represented on UI

To examine the reliability and consistency of rule logic applied to CV V2.0, precast concrete sample models were evaluated according the CV. Figure 31 is the validation result represented on UI and Figure 32 is one generated in the HTML format. Proposed rule logic was used to define IFC CV V2.0 and to evaluate IFC instance file as shown in two checking results. The identified errors from CV testing consists of five types of errors: (1) Uniqueness, (2) Reference existence, (3) Semantic accuracy, and (4) General syntax checking.



IfcWallStandardCase (17)

- ▶ GUIDs (Operator: And)
- ▶ History (Operator: And) [OPTIONAL]
- ▶ Naming (Operator: And)
- ▶ CAD Layer (Operator: And)
- ▶ Spatial Containment (Operator: And)
- ▶ Classification (Operator: And) [OPTIONAL]
- ▶ Objects Assignment (Operator: And) - [FAIL]
- ▶ RelatingGroup Assignment (Operator: And) - [FAIL]
- ▶ Material (Operator: And)
- ▶ Association Type (Operator: And)
- ▶ Product Local Placement (Operator: And)
- ▼ Path Connectivity (Operator: And) - [FAIL]

Instance	Structure	Constraints
#1082	\IfcRelConnectsPathElements	+
#1107	*	+
#1131	*	+
#1156	*	+
#52935	\IfcRelConnectsPathElements	+
#52960	*	+
#52984	*	+
#53422	*	+
#87917	\IfcRelConnectsPathElements	+
#87942	*	+
#87966	*	+
#88040	*	+
#118752	\IfcRelConnectsPathElements	+
#118777	*	+
#118801	*	+
#118825	\IfcRelConnectsPathElements	+
#118899	*	+

- ▶ Spatial Structure (Operator: And)

▼ Slab Type (Operator: And) - [FAIL]

Instance	Structure	Constraints
#111	*	+
#188	*	+
#237	*	+
#286	*	+
#335	*	+
#384	*	+
#433	*	+
#482	*	+
#531	*	+
#580	*	+
#629	*	+
#678	*	+
#727	*	+
#776	*	+
#825	*	+
#874	*	+
#923	*	+
#972	*	+
#1021	*	+
#1070	*	+
#1235	*	+
#3535	*	+
#52930	\IfcRelDefinesByType	+
#53409	*	+
#62550	*	+
#85834	\IfcRelDefinesByType	+

Figure 32 Validation report in the HTML format

Figure 33 represents checking about the reference relationship. This example shows #62551 instance, IfcStair lacks the reference with the material. Since the material relationship is required, the instance missing this relationship is represented in red. Figure 34 is the error showing lack of IfcRelDefinesByType of IfcBeam.

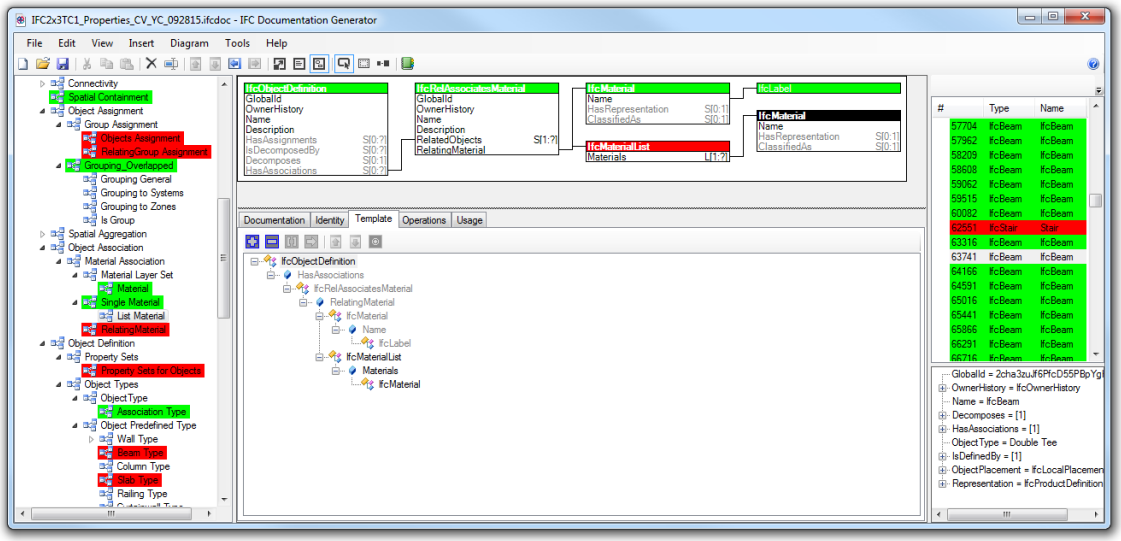


Figure 33 Lack of the material relationship

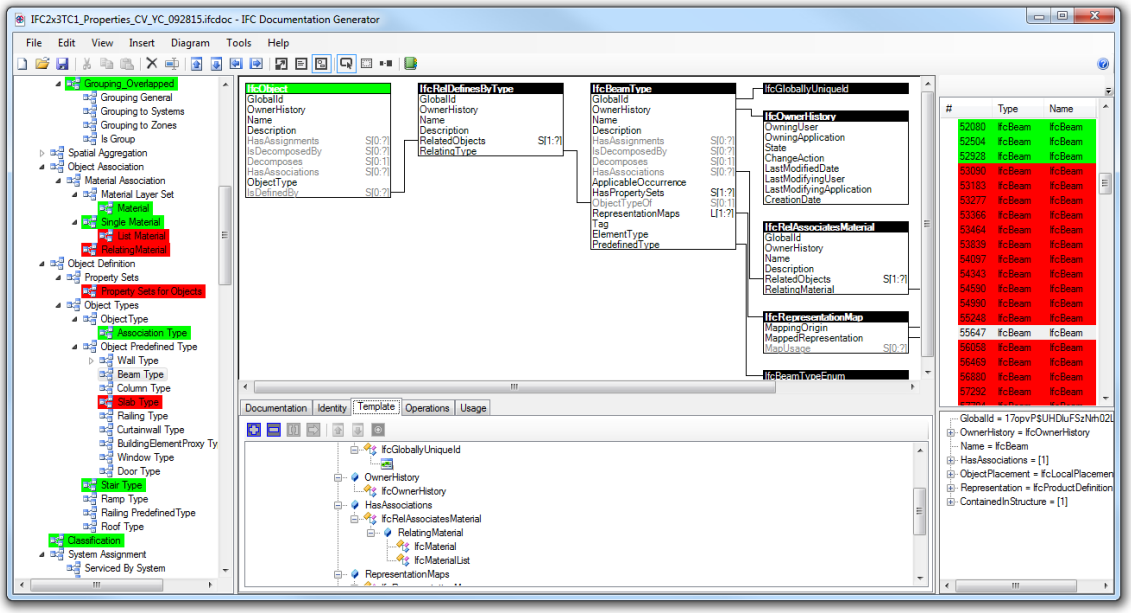


Figure 34 Lack of type reference

Figure 35 is the semantic accuracy and representation type defined in one attribute of a concept. The IfcBeam is only allowed to be represented by Curve2D using IfcTrimmedCurve and IfcPolyline and by SweptSolid using IfcExtrudedAreaSolid and IfcRevolvedAreaSolid. However, the beams in the IFC instance file do not fulfill this representation requirements for Items attribute and semantics for ContextOfItems of IfcShapeRepresentation.

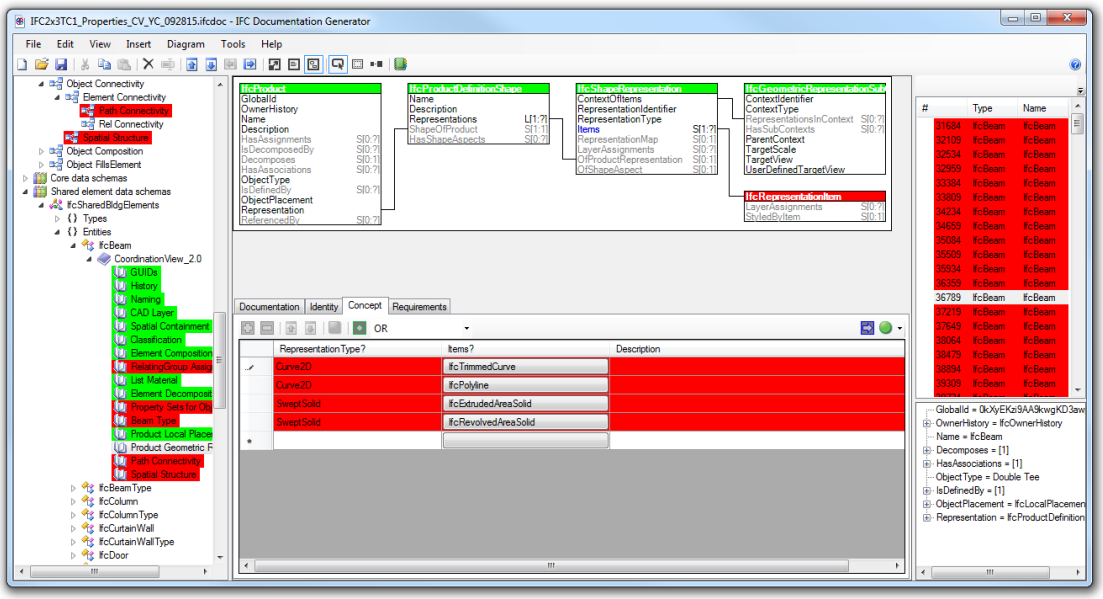


Figure 35 Semantic accuracy of representation type

Current GTDS certification process, which validate IFC instance files according to CV V2.0, is time-consuming, normally taking around six months (Eastman and Lee 2015). In addition, as discussed in Section 2, the process is hidden and implicit, and thus users and software vendors are not exposed to the validation process and criteria. On the contrary, this proposed validation process directly uses rule sets embedded in model views of the IfcDoc file and implement evaluation pertaining to syntactic and semantic specifications

in a few minutes. Thus, the proposed validation process is a way to not only utilize defined specifications of EMs as an active checking criteria, but also help domain experts and software developers automatically debug their IFC models in a short time according to exchange requirements. The CV V2.0 MVD and this validation process will be helpful to diverse software vendors who use it for developing IFC interfaces in that they can assess their IFC importing and exporting features according to CV V2.0 specifications.

(2) Construction Operations Building Information Exchange (COBie)

Since a number of building projects requires that facility managers involve in the early design phase, the requirements of facility management should be reflected and maintained during the design, construction, and FM phases. The Construction-Operations Building Information Exchange (COBie) is a model view for data exchanges regarding building asset information and facility management (FM) (William East, Nisbet et al. 2012). Diverse BIM authoring tools or applications such as the Autodesk Revit COBie Toolkit or FM Systems have IFC interfaces that can translate native objects into IFC ones according to the specifications of the COBie (Lee, Yang et al. 2015). Such IFC instance files provide project participants with necessary handover information demanded by facility managers or owners.

Thus, domain experts should evaluate BIM data to see whether it accurately contain required FM information defined in the COBie. To evaluate the applicability and extensibility of proposed rule logic of an MVD, this dissertation uses COBie specification for evaluating IFC instance files. In addition, this section shows the

redefined requirements of the COBie model view using the updated rule checking features of IfcDoc.

Based on the specifications of the COBie on the buildingSMART International website, the COBie IFC file was redefined on IfcDoc using rule checking features. Figure 37 shows 28 exchange models (EMs) of COBie V2.4 and Figure 36 shows 16 concept templates of this model view. Thus, 16 concept templates are iteratively composed and used to specify 28 EMs.

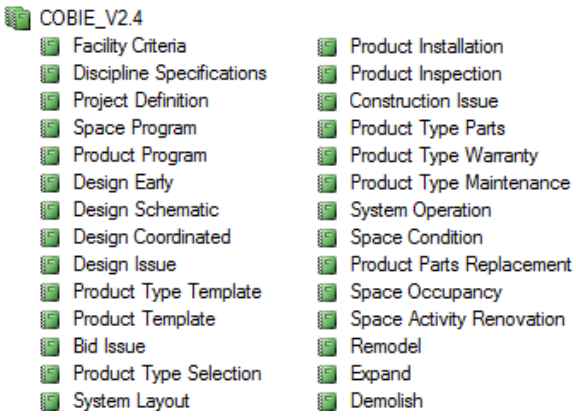


Figure 37 Exchange models of COBie V2.4 (Lee, Yang et al. 2015)

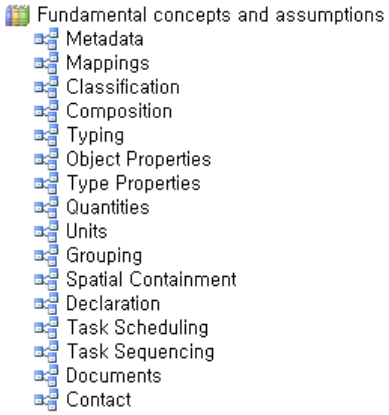


Figure 36 Concepts of COBie V2.4 (Lee, Yang et al. 2015)

The specifications and rule sets of the COBie were defined by the updated IfcDoc without having any extension of rule checking features. The primary types of rules embed uniqueness, semantic accuracy, relational references, and general syntax checking. Figure 38 shows an example of rule definition with regard to uniqueness. The Metadata concept

requires that the name attribute of IfcRoot has a unique value within BIM models throughout an entire project. The unique name of objects helps generate efficient data references of external data sets such as spreadsheets (Lee, Yang et al. 2015).

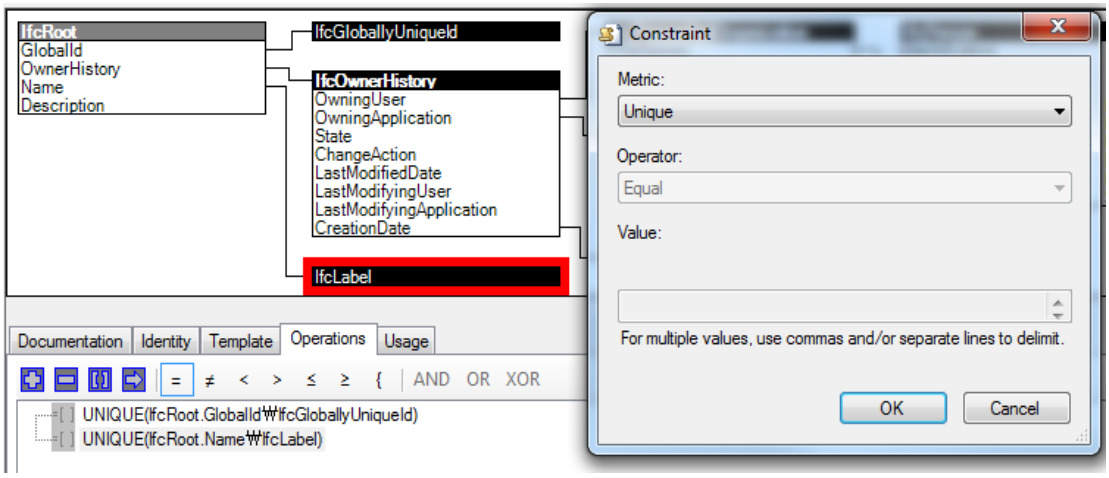


Figure 38 Uniqueness checking of an object name (Lee, Yang et al. 2015)

Figure 39 shows an example of accuracy checking of semantics. Since the COBie refers to the OmniClass taxonomy of CSI, asset classification should satisfy values of Source, Name, and Tokens parameters that can enumerate several semantic and entity options for an attribute.

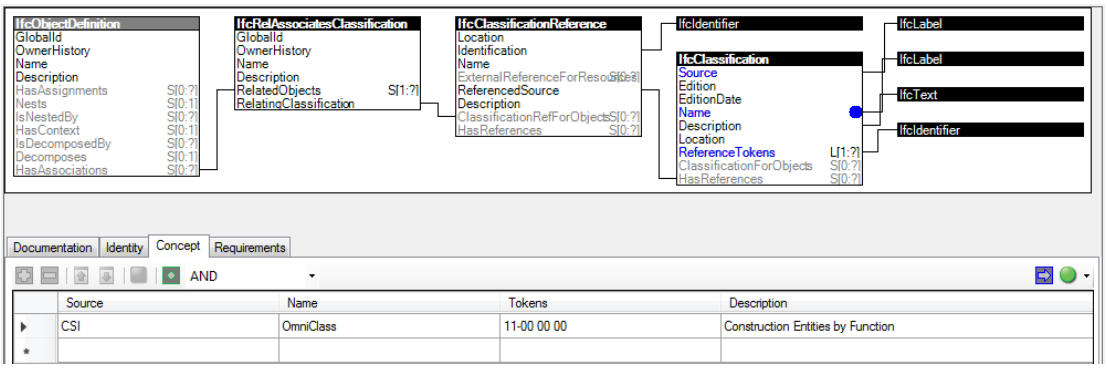


Figure 39 Semantic accuracy checking (Lee, Yang et al. 2015)

A relational reference checking is a primary rule type in the specifications of COBie. Figure 40 shows the relationship between spaces and a building level. The composition template has IfcObject that is inversely related to IfcRelAggregates. This IfcRelAggregates has a parameter for RelatingObject that allows IfcBuildingStorey in this concept. To define a hierarchical composition, this concept is assigned to IfcSpace so that an IFC instance file can have relationship between IfcSpace and IfcBuildingStorey.

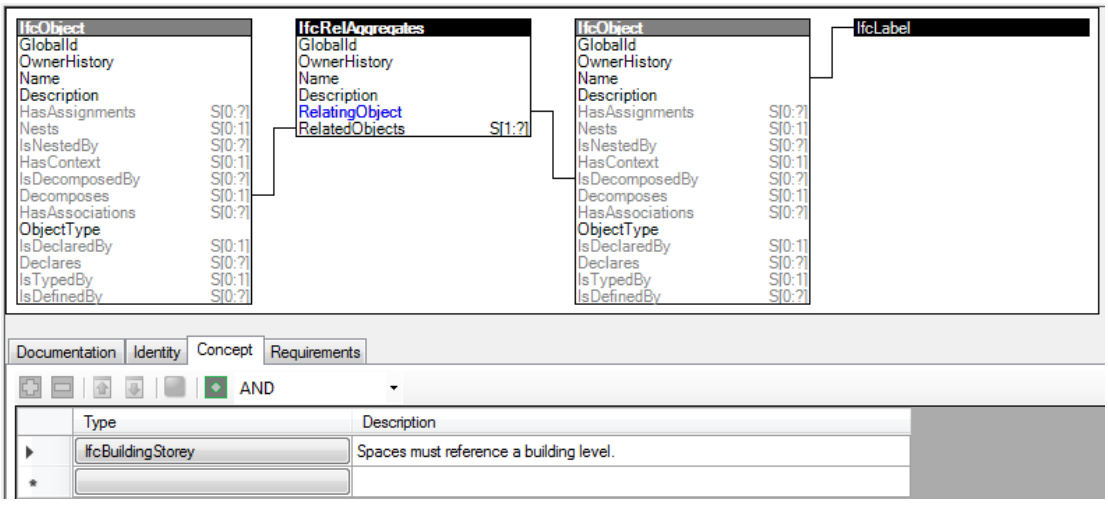


Figure 40 Referential relationship checking (Lee, Yang et al. 2015)

To evaluate the checking capabilities of IfcDoc according to the redefined COBie, IFC instance files embedding facility and asset management information were evaluated according to the COBie model view. Figure 41 shows one sample file, a clinic building that has rooms for a doctor, a nurse, a patient, operation, and X-ray, an information desk, a restroom and an HVAC. In addition, it contains furniture, fixtures, and equipment (FFE) such as a nurse call and has project requirements with regard to project participants, manufacturers, object and equipment quantities, and schedules. Since these design and project data are defined and reflected to a building design during the early design phase, they should be maintained throughout the entire project phase and passed over to facility managers (Lee, Yang et al. 2015).

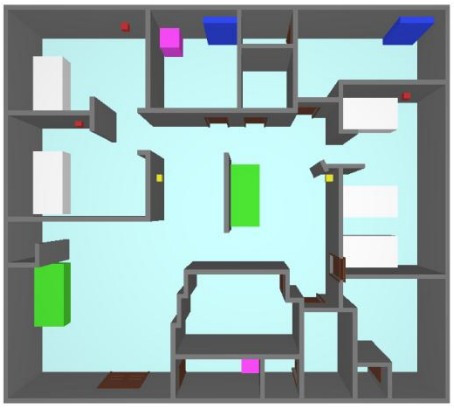


Figure 41 A clinic IFC model (Lee, Yang et al. 2015)

Figure 42 shows the UI validation report and Figure 43 shows the HTML one. As shown in the UI report, four concepts, Metadata, Mappings, Typing, and Spatial Containment, have items failed in the validation. By clicking instances listed on the right-hand pane, users can figure out the location and cause of a reported error. The validation outcomes provided instance files that have inaccurate semantics and relationships.

In summary, the evaluation with the COBie model view approved that the validation features added on IfcDoc are working correctly for addressing the specifications of the

COBie as expected. Thus, the checking frameworks are expected to resolve the problems of binding assignments associated with the COBie. In addition, project participants can ensure the interoperability of FM data embedded in BIM models. Hence, the redefined specifications and rule sets of the COBie IfcDoc file would provide software and domain experts with an automated rule-based checking process.

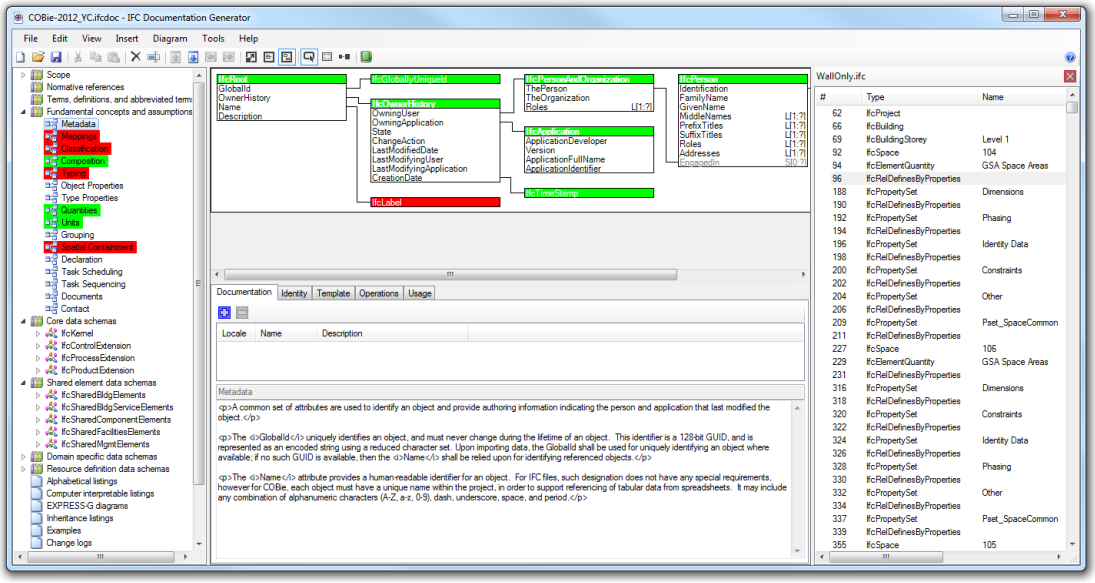


Figure 42 Validation report in the user interface (Lee, Yang et al. 2015)

Validation Results

Instance File	C:\YongCheol\ClinicSample.ifc
Project File	C:\YongCheol\COBie-2012_YC.ifcdoc
Model View	COBIE_V2.4
Exchange	Space Program
Tests Executed	14
Tests Passed	3
Tests Ignored	2
Tests Percentage	35%

IfcSpace (17)

▶ Metadata - [FAIL]
▶ Classification [OPTIONAL] - [FAIL]
▶ Composition
▼ Quantities [OPTIONAL]

Instance	Type	Name	Structure	Constraints
#92	IfcQuantityArea	GrossArea	+	+
#227	IfcQuantityArea	GrossArea	+	+
#355	IfcQuantityArea	GrossArea	+	+
#488	IfcQuantityArea	GrossArea	+	+

IfcBuildingStorey (1)

▶ Metadata - [FAIL]
▶ Mappings - [FAIL]
▼ Composition

Instance	Type	Structure	Constraints
#69	IfcBuilding	+	+

IfcBuildingElementProxy (16)

▼ Spatial Containment

Instance	Space Name?	Element Name?	Structure	Constraints
#112421	Patient room	Fire Alarm	+	+
	Patient room	Patient Bed	+	+
	Patient room	Nurse Call	*	
	Patient room	Faucet	+	+
	Patient room	Book Shelf	*	
	Patient room	Room Chair	+	+
#1112421	Patient room	Fire Alarm	+	+
	Patient room	Patient Bed	+	+
	Patient room	Nurse Call	*	
	Patient room	Faucet	+	+
	Patient room	Book Shelf	*	
	Patient room	Room Chair	*	
#2112421	Patient room	Fire Alarm	*	
	Patient room	Patient Bed	*	
	Patient room	Nurse Call	+	+
	Patient room	Faucet	*	
	Patient room	Book Shelf	*	
	Patient room	Room Chair	*	

Figure 43 Validation in the HTML format (Lee, Yang et al. 2015)

5.2.2 Validation of diverse IFC instances exported from software companies

This evaluation process, using diverse IFC instance files exported from Revit Architecture, Vectorworks, and Structureworks, helps confirm the consistency and ability of rule logic and its checking structure. In addition, the new approach for MVD validation will be provided to software vendors and professionals in the PCI industry so that they can employ this application in their system with regard to testing predefined rules and applying them in order to validate native models.

(1) Vectorworks

The IFC file exported from Vectorworks was validated against the EM1 exchange (Building Concept BC). Figure 44 shows the HTML report of validation of a precast concrete column model. Since the PCI-061 concept called precast piece material association has a requirement that IfcColumn refers to IfcMaterial as a RelatingMaterial attribute for a material association, the Vectorworks instance file that does not satisfy this requirement shows an error in the checking outcome.

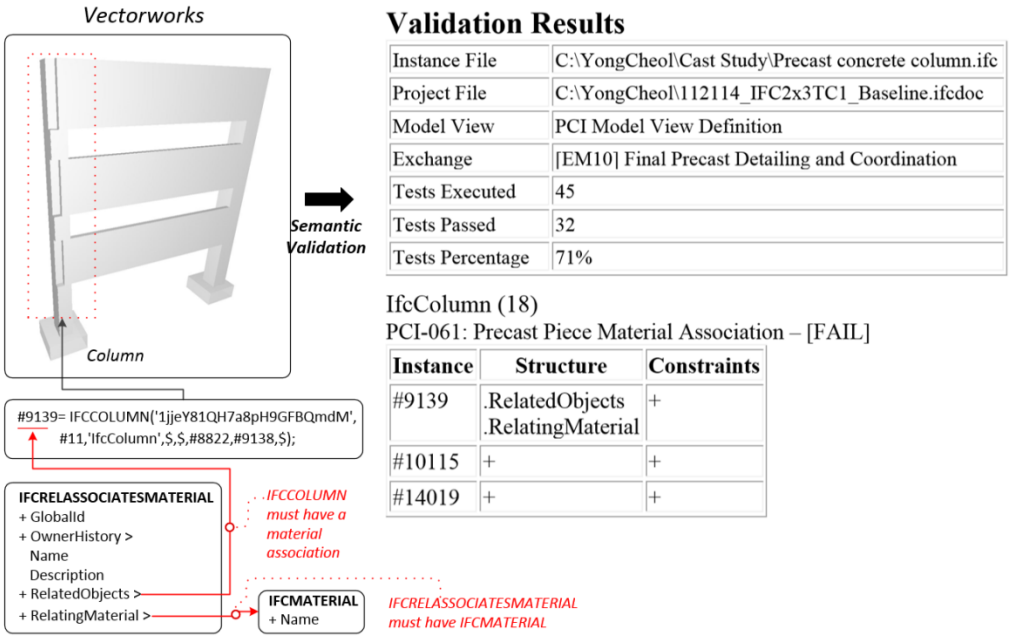


Figure 44 An HTML validation report of the IfcDoc pertaining to the PCI-061 precast concrete concept (Lee and Eastman 2015)

Figure 45 shows precast concrete model shown in the IFC viewer. This model consists of beams, columns, toppings, assemblies, reinforcing bars, meshes, and tendons. Figure 46 shows the geometry translated by IFC importer and exporter of Vectorworks. The model has skewed parts and lacks cobels and some rebars. In addition, some parts are missed while it is translated.

Object	Number
Beam	13
Column	8
Topping (BUILDINGELEMENTPART)	2
Assembly	21
Rebar	50
Mesh	20
Tendon	8

Figure 45 Objects before translation

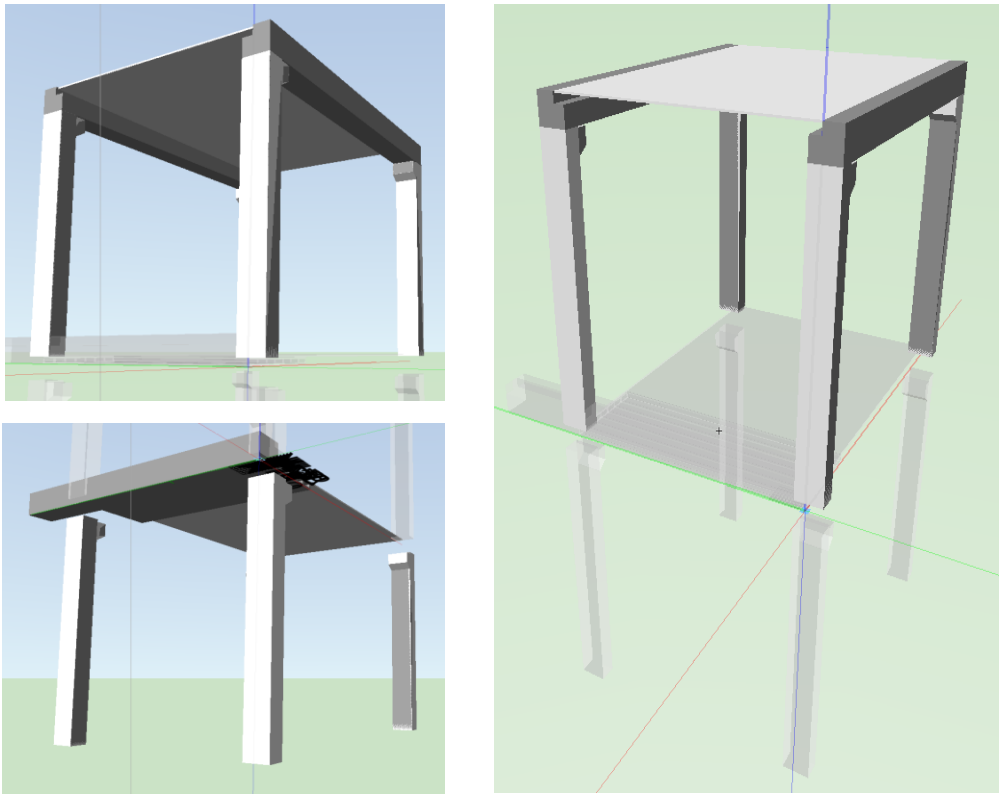


Figure 46 Importing the IFC file into Vectorworks and exporting it again

Figure 47 and 48 shows the number of objects changed and the altered geometry by translation processes, IFC importing and exporting. The incorrect translation says the IFC interfaces of Vectorworks have binding issues that result in these weird translations.

Object	Number	
	Before	After
Beam	13	5
Column	8	8
Topping (BUILDING ELEMENTPART)	2	1
Assembly	21	10
Rebar	50	0
Mesh	20	0
Tendon	8	0

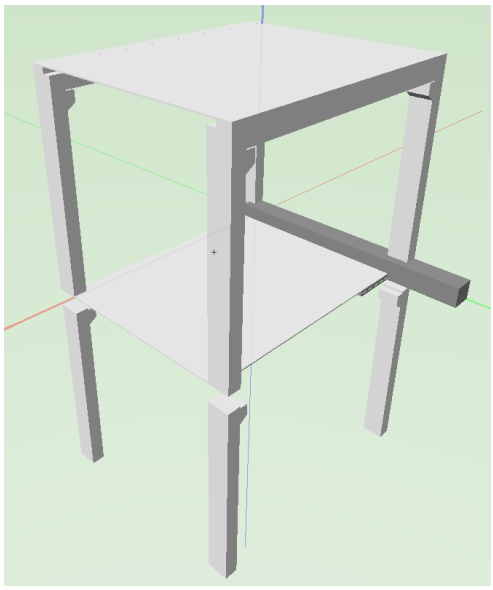


Figure 47 The number of objects and geometry changed after translations

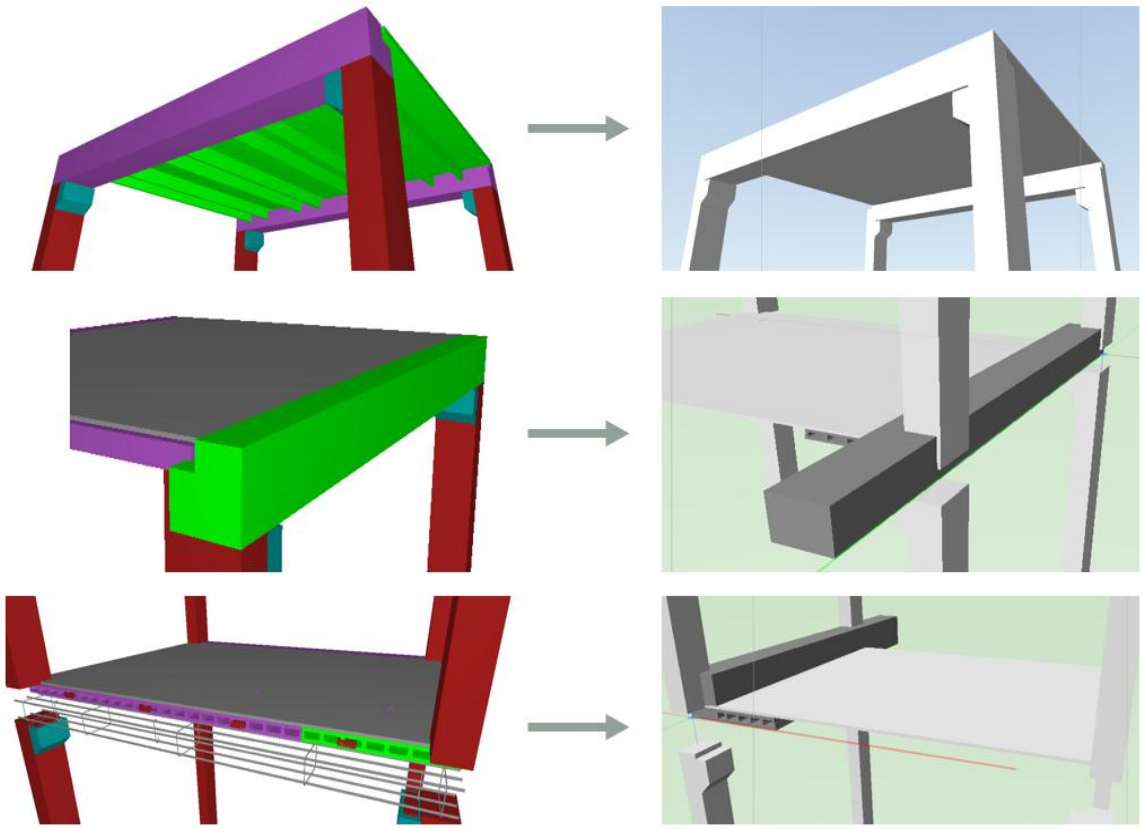
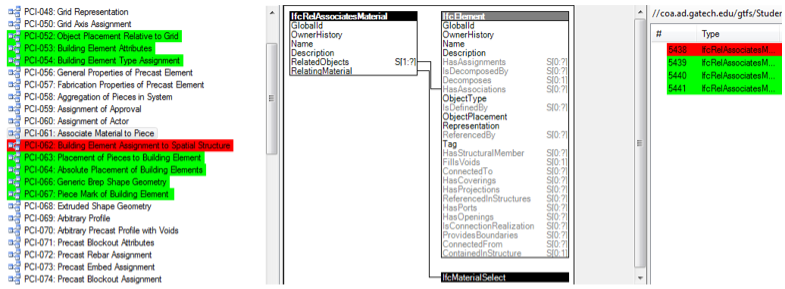


Figure 48 Geometry transformation (Lee and Eastman 2015)

The exported IFC file was evaluated by the IfcDoc application according to the PCI MVD in order to identify the causes of errors. Figure 49 and 50 represent examples of errors of this IFC model. The validation report shows that this model contained unnecessary entities, IFCGRIDAXIS into material reference and spatial structure relationships. The validation reports shows explicit errors and their locations so that users can debug errors and fix them easily. After fixing them, some objects missed were shown. Software vendors are expected to keep track of this binding errors in material references and spatial structure relationships and fix IFC interfaces for translating their native objects into IFC ones more accurately.

- UI Report



- HTML Report

IfcRelAssociatesMaterial (4)

▼ PCI-061: Associate Material to Piece - [FAIL]

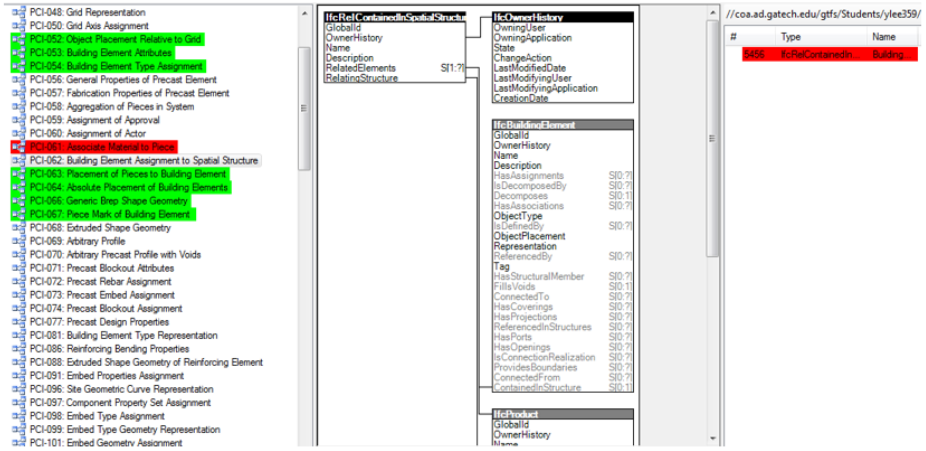
Instance	Structure	Constraints
#5438	.RelatedObjects\IfcElement	+
#5439	+	+
#5440	+	+
#5441	+	+

#5438= IFCRELASSOCIATESMATERIAL('3taLHh_IKHv9iZz6q4uvbE',#11,$,$,(#128,#254,#380,#530,#854,#1178,#1502,#3659,#3662,#3665,#3668,#3671,#3674,#3677,#3680,#3683,#3686,#3689,#3692,#3695,#3698,#3701,#3704,#3707,#3710,#4029,#4258,#4582,#4906,#5230),#5406);

#3659= IFCGRIDAXIS('UY-1',#1120,.T.);
#3662= IFCGRIDAXIS('UY-2',#1130,.T.);
#3665= IFCGRIDAXIS('UY-3',#1140,.T.);
#3668= IFCGRIDAXIS('UY-4',#1150,.T.);
#3671= IFCGRIDAXIS('UY-5',#1160,.T.);
#3674= IFCGRIDAXIS('UY-6',#1170,.T.);
#3677= IFCGRIDAXIS('UY-7',#1180,.T.);
#3680= IFCGRIDAXIS('VX-1',#1000,.T.);

Figure 49 Errors in material references: unnecessary entities, IFCGRIDAXIS

- UI Report



- HTML Report

IfcRelContainedInSpatialStructure (1)

▼ PCI-062: Building Element Assignment to Spatial Structure - [FAIL]

Instance	Structure	Constraints
#5456	.RelatedElements\IfcProduct	+

#5456= IFCRELCONTAINEDINSPATIALSTRUCTURE('3taLIb_IKHv9iZz6q4uvbE',#11,'BuildingStoreyContainer','BuildingStoreyContainer for Building Elements',(#128,#254,#380,#530,#854,#1178,#1502,#3655,#3659,#3662,#3665,#3668,#3671,#3674,#3677,#3680,#3683,#3686,#3689,#3692,#3695,#3698,#3701,#3704,#3707,#3710,#3828,#3950,#4029,#4258,#4582,#4906,#5230),#50);

Figure 50 Errors in spatial structure references: unnecessary entities

(2) Structureworks

With regard to validation of the IFC exporter of the Structureworks about IFC mapping correctness, the several IFC files were evaluated by the IfcDoc tool according to IFC coordination view 2.0 (CV) and the PCI MVD. IFC CV 2.0, which has an objective to facilitate coordination between the architectural, mechanical, and structural engineering task during the design phase (buildingSMARTInternational 2015), was employed to validate Structureworks IFC files since they were translated based on CV as the header section of IFC files shows (Table 25).

Even though these files were translated according to Coordination View (CV), it is suggested to allow users to bind Structureworks model data into IFC schema using the MVD of the Precast/Prestressed Concrete Industry so that numerous experts in PCI using this software solution[2] can exchange accurate data models and ensure the interoperability of building models.

Table 25 The header section of all IFC files of Structureworks

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION (('ViewDefinition [CoordinationView]'), '2;1');
FILE_NAME (", '2015-06-16T07:49:19', (' '), ('FinfrockDMC'), 'Structureworks
File Processor version 1.0', 'Structureworks BIM', ");
FILE_SCHEMA (('IFC2X3'));
ENDSEC;
```

[2] Structureworks website: http://www.structureworks.net/Overview.aspx

The IFC files of Structureworks were validated according to two exchange models (EM): EM 1 for Building Concept (BC) during the Preliminary Project Description phase and EM 10 for Final Precast Detailing & Coordination during the Fabrication and Erection phase. Validation using both Ems found out several types of issues:

- Missing attributes defined in the PCI MVD: EM 10 requires that IFCREINFORCINGBAR has IfcLabel and IfcReinforcingBarSurfaceEum even though the IFC 2x3 schema defines them optional.

IfcReinforcingBar (92)

▼ PCI-072: Precast Rebar Assignment (Operator: And) - [FAIL]

Instance	Structure	Constraints
#11315	\IfcLabel \IfcReinforcingBarSurfaceEnum	+
#18373	\IfcLabel \IfcReinforcingBarSurfaceEnum	+
#18380	\IfcLabel \IfcReinforcingBarSurfaceEnum	+

Figure 51 Fail report showing the missing attributes

- Representation type: EM 10 defines that IFCREINFORCINGBAR must be represented by IFCSWEPTDISKSOLID.

Table 26 BREP Representation of IFCREINFORCINGBAR

```
#11288=IFCFACETEDBREP(#11287);
#11310=IFCPRODUCTDEFINITIONSHAPE("","",(#11311));
#11311=IFCSHAPEREPRESENTATION(#12,'Body','Brep',(#11288));
#11315=IFCREINFORCINGBAR('0ayULk4OvC8x008W5n0I0b',#1,'SingleBar_
SB12-
1<B130060403>','B130060403','SingleRebar',#11318,#18366,$,$,0.0127,0.0001
26676869774374,2.57613315365561,.LIGATURE.,$);
```

IfcReinforcingElement (34)

▼ PCI-088: Extruded Shape Geometry of Reinforcing Element (Operator: And) - [FAIL]

Instance	Structure	Constraints
#200	*	IfcReinforcingElement.Representation\IfcProductDefinitionShape.Representations\IfcShapeRepresentation.RepresentationType\IfcLabel { CompositeCurve,MappedRepresentation
#2896	*	IfcReinforcingElement.Representation\IfcProductDefinitionShape.Representations\IfcShapeRepresentation.RepresentationType\IfcLabel { CompositeCurve,MappedRepresentation
#2903	*	IfcReinforcingElement.Representation\IfcProductDefinitionShape.Representations\IfcShapeRepresentation.RepresentationType\IfcLabel { CompositeCurve,MappedRepresentation

Figure 52 Fail report showing the diverse representation type

- Semantic error: EM 1 defines that the IFC instances must have one of the listed ElementType for the ElementType attribute of IFCWALLTYPE and must have one of the listed ObjectType for the ObjectType attribute of IfcWall.

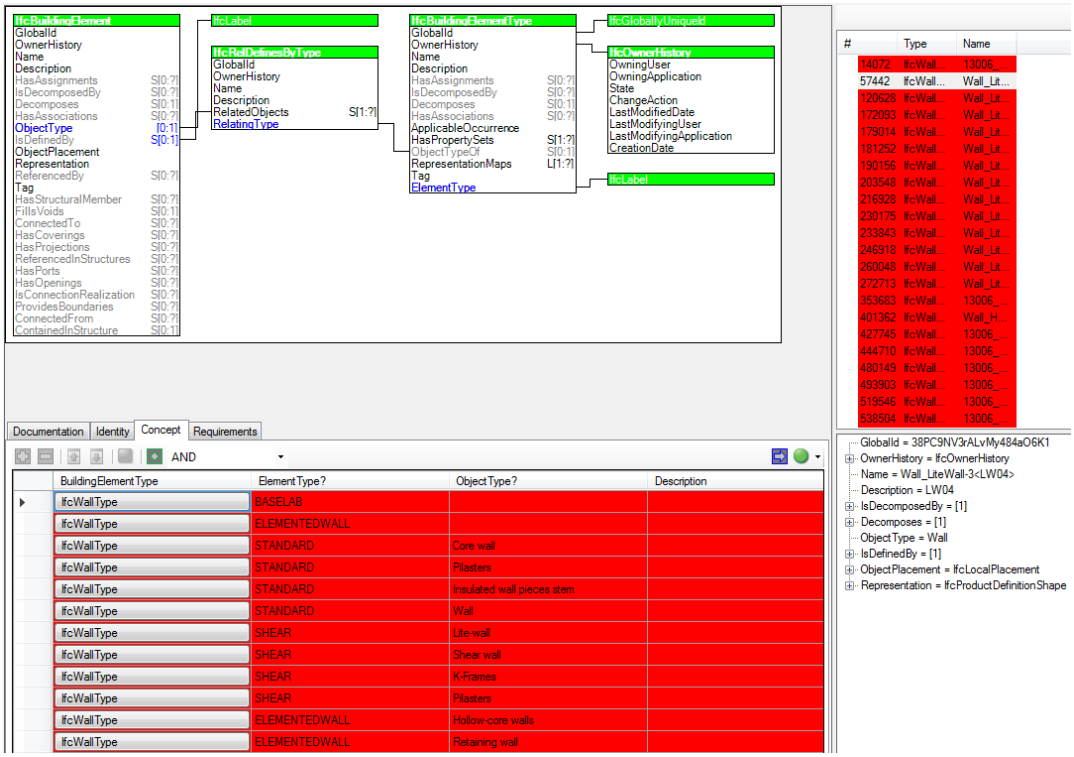


Figure 53 Semantic error of IfcWall of 13006-Structure_Default.ifc represented on user interface

- Syntax checking and invalid data type: Figure 54 shows syntax errors with regard to invalid data type of the NominalValue attribute of IFCPROPERTYSINGLEVALUE. In addition, NominalValue should be TYPE IfcValue = SELECT (IfcMeasureValue, IfcSimpleValue, IfcDerivedMeasureValue);

Validation Results

Instance File	C:\YongCheol\Project, Work\IfcDoc Extension\Structureworks_Validation\Structureworks\13006-Structure_Default(full).ifc
Project File	C:\YongCheol\Project, Work\IfcDoc Extension\Structureworks_Validation\072515_IFC2x3TC1_Properties_Baseline.ifcdoc
Model View	PCI Model View Definition
Exchange	[EM1] Building Concept
Tests Executed	44
Tests Passed	18
Tests Ignored	0
Tests Percentage	40%

This file contains format errors which may impact data validation results:

- #14539 .NominalValue - Unknown type: IFCNUMERICMEASURE(4.2418)
- #14540 .NominalValue - Unknown type: IFCNUMERICMEASURE(8.59790000000002)
- #14551 .NominalValue - Unknown type: IFCINTEGER(487)
- #14556 .NominalValue - Unknown type: IFCINTEGER(0)
- #14560 .NominalValue - Invalid Data: '1/1/1900 12:00:00 AM'
- #14561 .NominalValue - Invalid Data: '1/1/1900 12:00:00 AM'
- #14562 .NominalValue - Invalid Data: '1/1/1900 12:00:00 AM'
- #14563 .NominalValue - Invalid Data: '1/1/1900 12:00:00 AM'
- #14564 .NominalValue - Invalid Data: '1/1/1900 12:00:00 AM'

Figure 54 Syntax error of NorminalValue attribute of IFCPROPERTYSINGLEVALUE

Table 27 IFCPROPERTYSINGLEVALUE instances

```
#14560=IFCPROPERTYSINGLEVALUE('PlanDate','PlanDate','1/1/1900 12:00:00
AM',$);
#14561=IFCPROPERTYSINGLEVALUE('PourDate','PourDate','1/1/1900 12:00:00
AM',$);
#14562=IFCPROPERTYSINGLEVALUE('LoadDate','LoadDate','1/1/1900 12:00:00
AM',$);
#14563=IFCPROPERTYSINGLEVALUE('ShipDate','ShipDate','1/1/1900 12:00:00
AM',$);
#14564=IFCPROPERTYSINGLEVALUE('ErectDate','ErectDate','1/1/1900 12:00:00
AM',$);
```

5.3 Contributions

Design professionals are generally aware that the trend of building design is the use of BIM (Fischer & Kunz, 2004; Smith, 2008). In a collaborative BIM environment, ensuring the interoperability of building information models and employing the formal specifications of defining data exchange requirements are imperative. However, lack of software interoperability and functionality are rated as one of the greatest obstacles to the application of BIM (McGraw-Hill, 2009). In addition, end users and software vendors find it difficult to identify whether a design model has all of the required information according to the requirements of a given stages. As a result, domain experts require MVDs as indispensable specifications for defining their business regulations. Based on the critical success factors previously mentioned, an automated MVD checking process is a critical need for professionals.

To design an automated MVD checking process, this dissertation has two primary steps: (1) defining formal rule logic pertaining to the MVD and (2) developing rule-checking features on top of an open-source MVD generator, the IfcDoc tool, which is a stable and easy-to-use IFC validation method using open standards. Moreover, this research revealed several issues pertaining to MVD validation and definitions of the applicable scope of capabilities of the current mvdXML rules based on real-world scenarios. With regard to PCI domain, the PCI plays an integral role, significantly influencing other domains such as the steel, concrete, bridge, and brick industries. This validation framework of the precast concrete model view allows users to share and reuse its rule logic and assessment structure. One problem in defining such a model view is that model

view developers cannot easily track existing concepts and retrieve them for their purposes. In particular, as a result of this research, documented specifications are now implementable and electronically shareable among users who wish to employ defined data exchange specifications in the evaluation process of a P21 file in a particular exchange process.

The author expects this dissertation will significantly assist two parties: software vendors and domain professionals. Building information models have been widely employed throughout the design and construction industries, including the precast concrete industry. Furthermore, they use diverse types of building modeling authoring tools, and thus, need a neutral format for seamlessly exchanging building data. They use distinct BIM formats and formalized data exchange specifications and verify the accurate design of an instance file so that it meets exchange requirements and encompasses required data for particular exchange phases. To support this process, validation logic and structure will help software vendors evaluate their IFC interfaces, such as the importer or the exporter.

- This proposed dissertation and application for the automated interpretation and validation of MVDs is expected to help industry domain experts and software vendors determine if IFC instance models comply with the syntactic and semantic rule sets defined in MVDs requirements. Software vendors can automatically debug their IFC interfaces using the validation frameworks and keep track of the accurate locations of binding errors. In particular, while developing IFC interfaces, they iteratively evaluate IFC instance files translated by their IFC exporter or importer according to

the specifications of a targeted MVD. Based on the evaluation, they can assume the locations of errors in binding-assignment classes and fix identified errors of mapping native objects into IFC ones. In addition, I expect that domain professionals can utilize this validation approach during the design phases so that they can ensure that a design model has mandatory information demanded by other domains. Because of a number of growing requirements of a building design and project, they cannot be easily satisfied. Thus, an MVD that embeds the requirements of data exchange defined by a particular domain can play a pivotal role as criteria for design checking. Even though some of domain experts are not familiar with IFC, they are able to ensure the interoperability of BIM data exchange according to syntax and semantics. In particular, during the schematic design and design development phases, they evaluate their design according to specifications of a targeted MVD that represent required design and project information for data exchanges of a particular domain.

- Users are able to identify and modify the specification of concepts and rules. The existing GTDS system does not provide the details of concepts and rules, but only allows importing IFC files and validating against a selected MVD. However, the IfcDoc enables users to generate concepts and rules for their specification. In particular, the user-friendly reporting system generated in the HTML format and in the UI helps software vendors debug errors and find omissions in the binding process of their products.

- Formalized rule logic is the extent to which validation rules and checking processes are consistently followed, managed, and extended by other researchers or domain professionals. In addition, these logic is machine readable and its application provides a reasoning process such as mathematical induction and relational inference. Such generalization of rule types and checking frameworks would enable more effective inquiry of MVD knowledge that can declare standard for defining data exchange requirements throughout the MVD processes. The formal classification of rules of the PCI MVD that are implementable and extensible knowledge would also allow users to focus on necessary exchange constraints, which saves significant time and effort in developing MVDs. In particular, such a rule algorithm and structure formalizes the MVD knowledge and facilitates its integration with a native binding process, efficiently supporting MVD validation. In particular, generalized logic for MVD validation is expected to improve the process of model view validation. Users will reuse predefined concepts to fulfill multiple MVDs without having to modify the concepts. Such rule logic plays a pivotal role as a framework for defining and executing rules on the IfcDoc. Moreover, the IfcDoc helps share concept descriptions and rule sets among same domain experts. The IfcDoc application is open to the public, and thus, an ifcdoc file or an mvdXML file that encompasses user-defined exchange specifications can be easily shared so as to discuss and communicate the well-formedness of concepts and the MVD. In addition, the IfcDoc application supports reusing the predefined concepts and rules using the import feature. The concepts for exchanges and their rule sets can be reused for the similar domain

exchanges. The initial objective of concepts is to facilitate reusing the existing concept descriptions for similar MVDs.

- Even though the IfcDoc application is in development using PCI concepts, the tool would be used to certify the IFC interface such as an exporting feature on software vendors and industry domain experts. For the new versions of IFC schema files, MVD is mandatory to validate the binding process between the IFC schema and a native model. In addition, the MVD mapped P21 file must be validated to evaluate whether the file has appropriate data defined in implementation agreements of concept templates. This checking environment for MVD instance validation using the IfcDoc with buildingSMART in the U.S would provide the specific rule sets for the new release of the future IFC schema so that professionals validate their IFC interfaces with regard to the conformity of MVDs.

The processes of automated MVD validation can accelerate the use of BIM and facilitate the process of current design and construction work for the AEC industries. This opportunity would offer the values of greater regulatory predictability and consistency, reduce human errors in a design process, and lead to improve design quality. Hence, the research related to the automated process of a design would bolster the adoption of BIM and its accessories starting during the predesign phase, collaborating with the practitioners at the government organization and the industry.

CHAPTER 6 CHALLENGES, DISCUSSIONS, AND CONCLUSIONS

6.1 Challenges and limitations

The suggested rule logic and framework were evaluated by implementing rule sets for validating a P21 file pertaining to the PCI MVD and by promoting a proposed application for software vendors such as StructureWorks, Vectorworks, and Autodesk. This validation application was also used to evaluate a P21 file according to IFC CV 2.0 and COBie. In addition, MVDs in other domains such as the MVD of the American Concrete Industry (ACI) and the American Institute of Steel Construction (AISC) would be evaluated by this application in the future. While identifying rule logic from a number of concept descriptions and validating various IFC instance files according to several model views, the author was able to discover several challenging issues and limitations in defining rule sets and executing MVD validation.

(1) IFC schema structure developed for data exchange

The IFC schema has been designed and developed as a neutral format for data exchange, not as a particular format for data modeling (Lee and Eastman 2015). To embrace diverse domain knowledge and represent various design data, it has been edited several times: the current version is the IFC 4. Since it is open to various definitions and interpretations, rule types for some checking cases of attributes and references cannot be strictly defined. In other words, all possible values and relations should be considered and defined in formalizing rule types.

(2) Scope of MVD validation

The scope of MVD checking is another issue that requires further discussion because the scope of defining a model view has not been determined. Even though diverse domains have defined MVDs, no standard for defining the specifications of data exchanges exists. Thus, required data types and rules cannot be consistently declared in concept templates. For example, the scope of validation pertaining to uniqueness, including GUID and SET, and that of compliance with the IFC schema such as OwnerHistory are slightly ambiguous. Even though these rules must be fulfilled on the syntax level defined in the IFC schema and the EXPRESS language, such requirements are necessary in an exchange of building data. Present validation approaches, however, are too fragmented to evaluate an IFC instance file in terms of syntactical integrity, semantic accuracy, and programming requirements. In this situation, which does not involve an integrated approach that addresses three types of validation, we consider the IFC schema-associated evaluation a relatively problematic issue.

As another example of the limitation of the scope of validation, the MVC-818 concept, referred to as "Face Based Surface Model," states that the ContextOfItems attribute of IfcShapeRepresentation is required as a type of IfcRepresentationContext. This requirement, however, is the same rule defined in the IFC schema 2x3. In other words, this rule does not have to be included in MVDs. Furthermore, IfcProductDefinitionShape inversely refers to IfcProduct through ShapeOfProduct. This relationship means that all IfcProduct and its subtypes should follow this type of shape representation. However, since the covering of finishing elements such as cladding and flooring uses the "Face-

Based Surface Model", the MVC-818 concept should be assigned only to IfcCovering instead of IfcProduct. This example shows that to bind native models to IFC objects, specific entities should delimit the applicable area.

The scope of MVD validation also entails several other limitations that relate to the semantics and syntax of objects. With regard to the former, any concept with a requirement for a semantic cannot be applied to all entities because IFC interfaces have heterogeneous mapping processes that generate distinct values for IfcLabel. Thus, removing attributes and their parameters for requiring semantic values such as IfcLabel, IfcText, and IfcIdentifier (PCI-053 ObjectType) would be more reasonable. With regard to the latter, syntax, the scope of checking is obscure because even though MVD is a subset of the IFC schema, a concept that definitely follows the same requirements of the IFC schema is worthless in validation. Thus, excluding a concept that has only syntax requirements defined in the IFC schema such as "PCI-047: Grid Name" would lead to an efficient validation process.

At the level of the model view definition at the IfcDoc platform, the model view must include all entities, attributes, and relationships that are expected to be used in exchange processes. However, from the perspective of model view validation, both required and optional rules are associated with the existence of objects. Regardless of rule satisfaction, optional checking should be skipped. IfcDoc has an option to define this condition, but if several constraints are combined into one checking process, neither required nor optional checking can easily be managed.

(3) Several constraints and their priority for checking sequence

Mandatory and optional checking can be interpreted and implemented from various perspectives based on its applied placement. This checking type is used in three cases: an exchange, an attribute, and a cardinality. In these cases, checking consists of two aims: existence checking or pass-required checking. For example, when a user defines an IfcWall concept that is mandatory for a specific data exchange such as that between an architect and a constructor, if an IFC instance file does not have the IfcWall entity, we must know if we can report it as an error or not. Because it is not clear whether we can define a rule that an IFC instance file must have a specific object, this issue is closely related to the scope of MVD checking. When the mandatory setting is applied to an attribute and a cardinality, if an entity lacks either one, we can report such a case as an error because their existence can be asserted and evaluated.

One big challenge is the sequence of multiple rule executions. As the implementation of one attribute checking generally requires several rule types, rules should be well-organized and logically ordered to evaluate complex relationships and numerous requirements of one attribute validation. The priority of rules, however, currently cannot be defined and the checking sequence cannot be managed efficiently. The unexpected implementation of rule types in a wrong checking order could generate inconsistent and unintended validation outcomes. As an example, property-set checking requires multiple rule types to evaluate an IFC instance file according to the accurate property-set and corresponding property single values. To address such numerous rules, conditional checking using a parameter allows users to declare the order of two rule types. For

example, if a value of a NAME attribute of an IFC instance file satisfies one defined in a property-set concept, checking for property single values can be implemented. The sequence of executions of multiple rule types is still one of concerns that should be carefully addressed. One technical problem of IfcDoc resides in checking the optional and mandatory setting of IfcPropertySingleValue. An IFC instance file can satisfy the parts of optionally defined IfcPropertySingleValue, but the current version of IfcDoc cannot handle this issue. The IfcDoc source code would be manually updated for addressing this issue.

(4) Diverse approaches to defining the MVD and validation

The National Building Information Modeling Standard (NBIMS), which describes all MVD documentation processes, suggests formal approaches to defining concepts and the MVD. Other approaches have defined them in another format: a concept block, a concept template for the IFC 2x3 schema, and a restructured concept template for the IFC 4 schema. Various methods have generated distinct structures and contents of concepts, resulting in inconsistent rule types and implementation (Lee and Eastman 2015). This dissertation uses the IfcDoc v9.3, which has been approved as the official method of defining MVD by bSI. If new method for defining model views is proposed and used, the concept templates and embedded rules will also change. As described above, software vendors are able to interpret the IFC schema distinctly or lack the ability to represent necessary information. Thus, the IFC schema has been updated to efficiently represent design information. This problematic issue influences the development of rule logic.

In terms of PCI MVD, since the modularized concept was first applied, the scope of concept application has been so vague that users have not been able to define the configurations or formats of a concept of model view consistently. PCI concepts have been defined as a concept block with specific information that targets clear objects. However, the IfcDoc adopted a way of using a concept template so that the broad definition and requirements are used in several objects. In the case of the PCI MVD, this difference in approach caused considerable revision of existing PCI concept blocks. Now, the PCI MVD combines a concept block and a concept template. Thus, while some concepts focus on very specific objects, others address a broad area that misses a specific target object. This vague scope can lead to errors when a large sample with diverse types of entities is validated. Thus, these approaches, which we have used to define concept descriptions, entail substantial changes in methods of defining model views, their procedures, and their forms. This dissertation uses the IfcDoc v9.3, which is officially approved as the method of defining the MVD by bSI, so it has a rule-checking process embedded in the specific method of defining an MVD. If another approach is officially applied to defining the IFC schema and the MVD, the concepts and associated rules, including rule logic, will also change.

(5) Integrity of formalized and generalized rule logic

Not all being is absolute perfection. Imperfection of generalized logic and statements for the proposed validation approach is a critical limitation. Formalization and generalization subject to address diverse practical cases in several industry domains must be carefully addressed. Because of diverse interpretations and perspectives of generalized

rule logic and checking frameworks, the suggested rule logic cannot fulfill demands of the specifications of other domains or implement exceptional checking cases of BIM data exchange. The author acknowledges that even though the suggested rule logic has been refined according to a number of practical evaluations with multiple IFC instance files and three model view definitions, the formalized rule logic and checking processes contains domain-oriented constructs and validation steps. To refine such limitation and improve integrity of generalized rule logic, this dissertation should be shared and evaluated by diverse domains and BIM authoring tools.

(6) Topology checking and local placement

Because model view validation mostly addresses semantics, topological evaluation is not yet implementable. A semantical match with a defined model view can fulfill data exchange requirements, but it does not guarantee the physical accuracy and the integrity of instance files. For example, although checking if wall objects are correlated to corresponding slabs or if facilities and equipment are properly located in an associated space is indispensable, it is not supported by present logic or applications. We could resolve such issues by an algorithm that analyzes IFC instances.

6.2 Discussions and Conclusions

With the growing requirements of a building design, BIM data will integrate a significant number of requirements and project specifications demanded by various stakeholders (Lee and Eastman 2015). Product models must include accurate project information and design data required by domain experts and stakeholders in a particular

project stage (Eastman 1999). In addition, they have to ensure interoperability of BIM data according to an MVD including the domain-oriented specifications of data exchanges. Even though several MVDs have been defined for specifying data exchange requirements for diverse domains, BIM data exchanges using IFC still have numerous problems and challenges in syntax and semantics. To identify and debug such errors, users manually examine BIM data because of a lack of a robust approach for MVD validation (Lee, Eastman et al. 2015; Solihin, Eastman et al. 2015). To ensure interoperability, rule logic were categorized from the specifications of the PCI MVD and implemented on the IfcDoc tool. The proposed validation logic and structures are a unique approach to enabling software vendors and end users to debug the IFC binding process of native model objects and ensuring the interoperability of the data exchange of BIM data. Ultimately, all the data in an exchange model or the MVD resides in one or a set of variables in the data structure, that is, the basis for what validation must address. The validation framework contains types or rules and generalized implementable specifications in concept descriptions that are executed by diverse parameters and checking features.

With regard to implementation of this validation process, syntactic and semantic checking compared the values and the relational structures of instance files with predefined criteria such as the IFC schema, the EXPRESS language, and model view requirements. In addition, we have no explicit standard or uniform framework in defining modularized concept templates. In an effort to accomplish this objective, the author provided bSI with the identified rule logic and framework for developing an

intuitive checking environment, IfcDoc, for syntax and MVD validation. This application is expected to help project participants evaluate semantics of BIM data according to the specifications of the MVD but also implement syntactic checking with regard to compliance with the IFC schema (Lee, Eastman et al. 2015).

As discussed in the Challenges and Limitations section, the formalization of the specifications of model views cannot be perfectly accomplished. However, in an effort to formalize rule sets of an MVD, this validation logic and frameworks would be a stepping stone for developing a robust validation process and extending rule sets associated the requirements of BIM data exchanges. In particular, the suggested approach to evaluating BIM data exchanges would be helpful MVD developers to consistently figure out what information can or cannot be contained in the specifications of an MVD.

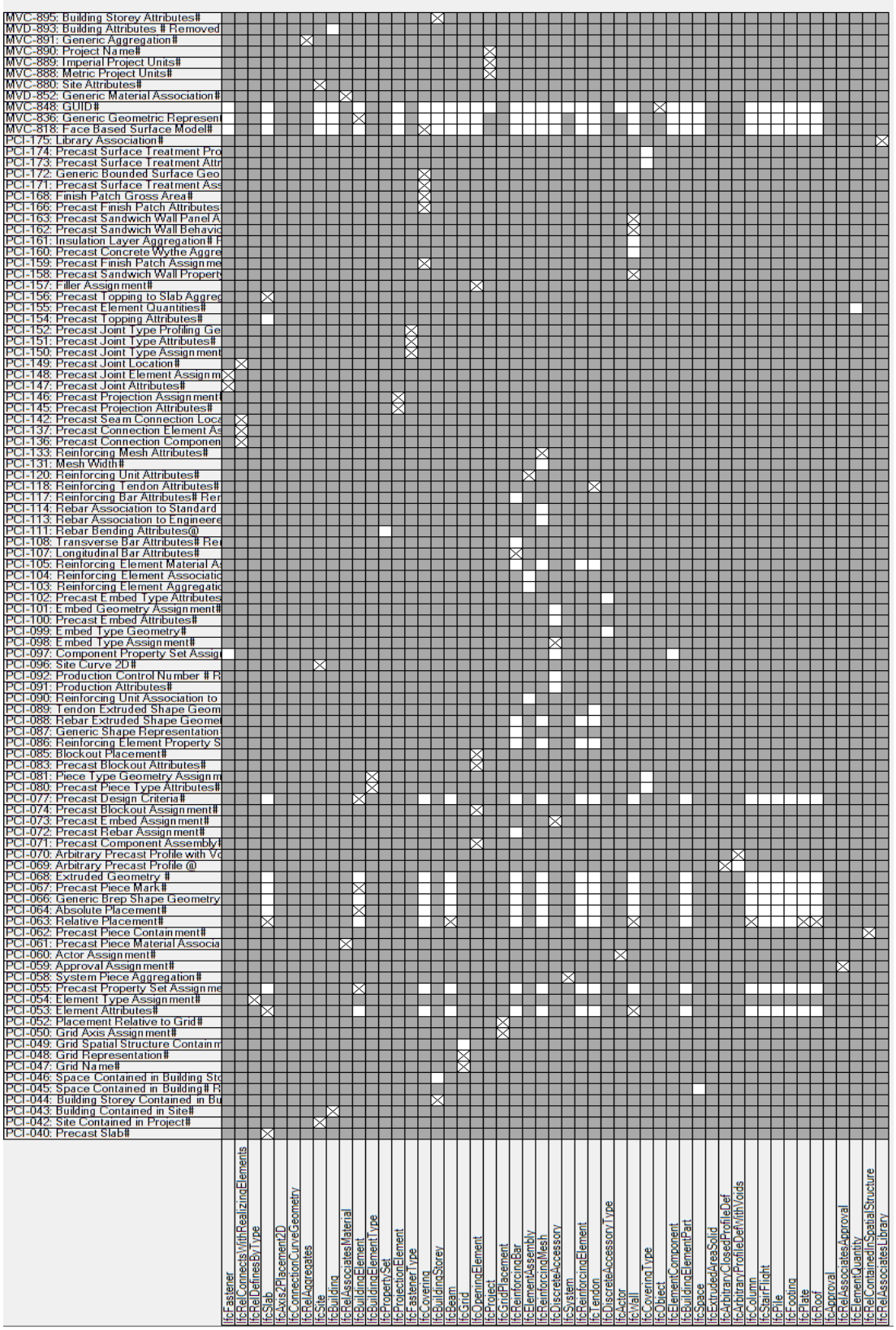
As a next step, a Georgia Tech research team in the Digital Building Laboratory will conduct research regarding model view definitions of steel, concrete, and bridge domains and update additional rule logic that can address exceptional cases. In addition, this research will contain an engineering ontology for PCI MVDs that can formalize MVD knowledge and help generate an explicit link between concept rules and formal information models that can reduce effort and time at defining MVDs and their rule sets and validating the IFC instance files. Logically organized rules in accordance with an ontology model will allow the concept modules to be logically defined for their reuse in various domains. This research will address five topics: (1) how accurately all model view requirements are defined for fulfilling the specifications of one domain, (2) how

modules are generated and organized to improve their reusability, (3) how modules are efficiently assigned to entities and how they can avoid redundancy of concept use, (4) how users can identify if defined MVDs contain all requirements agreed in the IDM process, and (5) how BIM models can be validated logically according to defined model views. Thus, to provide a concrete framework of model view definition and validation, this dissertation will extend the ontology-integrated research that addresses the scope of knowledge and criteria using the semantic web and reasoning processes. In addition, this extended research will provide an extensible and shareable data set for reusing MVDs in diverse disciplines. To integrate several validation processes to ensure the interoperability and quality of product model data, industries, academia, and governments must collaborate to formalize specifications of data exchanges and standardize a guideline of model view definition that will lead to a consistent and reliable validation process for BIM models and ultimately, accelerate the use of BIM in the architectural, engineering, construction, and facilities management industries.

APPENDIX A- Geometric Representation Mapping Table

Shape Types		EM-1	EM-2	EM-3	EM-4	EM-5	EM-6	EM-7	EM-8	EM-9	EM-10	EM-11A&B
Object Types	**Main Concept**											
Site perimeter	PCI-036	Composite Curve	<none>	Composite Curve	<none>	Composite Curve	Composite Curve	<none>	<none>	<none>	<none>	<none>
Site mesh		<none>	<none>	<none>	<none>	<none>	<none>	<none>	<none>	<none>	<none>	<none>
Building Element												
Slab	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Slab Aggregate		<none>	<none>	<none>	Composition	Composition		Composition	Composition		Composition	Composition
Wall	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Wall Aggregate		<none>	<none>	<none>	Composition							
Beam	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Beam Aggregate		<none>	<none>	<none>								
Column	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Column Aggregate		<none>	<none>	<none>								
Stair	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	B-rep	B-rep	B-rep
Stair Aggregate		<none>	<none>	<none>								
Ramp	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	B-rep	B-rep	B-rep
Ramp Aggregate		<none>	<none>	<none>								
Member	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Member Aggregate		<none>	<none>	<none>								
Element Proxy	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Element Proxy Aggregate		<none>	<none>	<none>								
Fillers	PCI-166	B-rep	B-rep	B-rep	B-rep	B-rep	Extrusion	B-rep	B-rep	Extrusion	B-rep	B-rep
Stair Flight	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep		B-rep	B-rep		B-rep	B-rep
Stair Flight Aggregate		<none>										
Ramp Flight	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep
Roof			B-rep	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep		B-rep	B-rep
Roof Aggregate	PCI-166	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition
Footing	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep	B-rep
Footing Aggregate		<none>	<none>	<none>	Composition	Composition	Composition	Composition	Composition		Composition	Composition
Pile	PCI-068.069	Extrusion	Extrusion	Extrusion	B-rep	B-rep						
Pile Aggregate		<none>	<none>	<none>	Composition	Composition						
Covering	PCI-170	<none>	<none>	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	<none>	Bounded Surface	**Bounded Surface**
Grid	PCI-048	Curve	Curve	Curve	Curve	Curve	Curve	Curve	Curve	Curve	Curve	<none>
Tendon		<none>	<none>	Swept Disk	Swept Disk	<none>	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk
Tendon aggregate		<none>	<none>	Swept Disk	Swept Disk	<none>	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk
Rebar	PCI-102, 103, 104	<none>	<none>	Swept Disk	Swept Disk	<none>	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk
Rebar Aggregate		<none>	<none>	Swept Disk	Swept Disk	<none>	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk	Swept Disk
Discrete Accessory	PCI-101	Composition	<none>	Composition	<none>	<none>	Composition	Composition	Composition	Composition	Composition	Composition
Connection	PCI-136	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition	Composition
Connection Aggregate		<none>	Composition	Composition								
Protusion	PCI-146	Projection	Projection	Projection	Projection	Projection	Projection	Projection	Projection	Projection	Projection	Projection
Hole/Pocket	PCI-074	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement	OpeningElement
Joint		<none>	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve	Composite Curve
Finish Patch	PCI-153	<none>	<none>	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	Bounded Surface	<none>	Bounded Surface	Bounded Surface
Topping & Wash	PCI-154	<none>	B-rep	B-rep	B-rep	<none>	B-rep	B-rep	B-rep	B-rep	B-rep	<none>

APPENDIX B- PCI Concepts and Exchange Models



REFERENCES

Anumba, Chimay J, Raja RA Issa, Jiayi Pan, and Ivan Mutis. "Ontology-Based Information and Knowledge Management in Construction." *Construction Innovation: Information, Process, Management* 8, no. 3 (2008): 218-39.

Autodesk Inc. "Bim and Facilities Management." http://www.arc-corporate.com/_pdf/bim_and_fm_whitepaperv3.pdf, Retrieved December 12, 2014.

Avigad, Jeremy; Donnelly, Kevin; Gray, David; and Raff, Paul (2007); "A formally verified proof of the prime number theorem", *ACM Transactions on Computational Logic*, vol. 9 no. 1

buildingSMART. "Ifc Certification 2.0: Specification of Certification Process ", 2010.

buildingSMART. "Ifcdoc - the New Mvd Development Tool." http://www.buildingsmart-tech.org/blogs/msg-blog/ifcdoc-the-new-mvd-development-tool, Retrieved March 8, 2015.

buildingSMART. "National Budiling Information Modeling Standard." (2007).

buildingSMARTInternational. "Concepts of the Ifc Coordination View " http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/concepts, Retrieved June 26, 2014.

buildingSMARTInternational. "Ifc Coordination View V2.0." http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/summary, Retrieved July 3, 2015.

Chipman, Tim, Thomas Liebich, and Matthias Weise. "Mvdxml V.1-Draft: Specification of a Standardized Format to Define and Exchange Model View Definitions with Exchange Requirements and Validation Rules." 2012.

Eastman, C, Jae-min Lee, Yeon-Suk Jeong, and Jin-kook Lee. "Automatic Rule-Based Checking of Building Designs." *Automation in Construction* 18, no. 8 (2009): 1011-33.

Eastman, C, and Yong-Cheol Lee. "Are Model Views Necessary?" In DBL White Paper. Digital Building Laboratory: Georgia Institute of Technology, 2015.

Eastman, C., Y-S. Jeong, R. Sacks, and I. Kaner. "Exchange Model and Exchange Object Concepts for Implementation of National Bim Standards." *Journal of Computing in Civil Engineering* 24, no. 1 (2009): 25-34.

Eastman, C., R. Sacks, I. Panushev, S. Aram, and E. Yagmur. "Information Delivery Manual for Precast Concrete." 2009.

Eastman, C., R. Sacks, I. Panushev, S. Aram, and E. Yagmur. "Precast Concrete Bim Standard Documents:Model View Definitions for Precast Concrete." (2010).

Eastman, C., P. Teicholz, R. Sacks, and K. Liston. Bim Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors. Wiley, 2011.

Eastman, Charles M. "Information Exchange Architectures for Building Models." *Durability of building materials and components* 8 (1999): 2139-56.

Eastman, Charles M., Yong-Cheol Lee, Kereshme Afsari, Shani Sharif, Donghoon Yang, and Samaneh Zolfagharian. "Precast Concrete National Bim Standard - Phase Iii-C." *Precast/Prestressed Concrete Institute (PCI)*, 2015.

Fowler, Julian. "Step for Data Management, Exchange and Sharing." (1995).

Froese, T. "Future Directions for Ifc-Based Interoperability." *Journal of Information Technology in Construction* 8 (2003): 231–46

Gallaher, M.P., A. C. O'Connor, J. L. Dettbarn, and L. T. Gilday. "Cost Analysis of Inadequate Interoperability in the Us Capital Facilities Industry." *National Institute of Standards and Technology (NIST)* (2004).

Ghang Lee, Charles M. Eastman, Rafael Sacks, Shankant B. Navathe. "Grammatical Rules for Specifying Information for Automated Product Data Modeling." *Advanced Engineering Informatics* (2006): 155–70.

Gnanarednam, Mehala, and Himal Suranga Jayasena. "Ability of Bim to Satisfy Cafm Information Requirements." Paper presented at *The Second World Construction Symposium*, 2013.

Hietanen, J. "Ifc Model View Definition Format." edited by International Alliance for Interoperability: buildingSMART, 2006.

Hietanen, J. and Final, S. "Ifc Model View Definition Format." International Alliance for Interoperability (2006).

Hodges, Wilfrid "Classical Logic I: First Order Logic", in Goble, Lou (ed.); *The Blackwell Guide to Philosophical Logic*, Blackwell (2001).

Howard, R, and B Bjo¨rk. "Building Information Modelling Experts' Views on Standardisation and Industry Deployment." (2007).

IAI. "Industry Foundation Class (Ifc) Data Model." BuildingSMART, 2003.

Digital Building Laboratory, "Precast Bim Standard Project." http://dcom.arch.gatech.edu/pcibim/, Retrieved June 17, 2014.

Lee, Jin-Kook, Charles M Eastman, and Yong-Cheol Lee. "Implementation of a Bim Domain-Specific Language for the Building Environment Rule and Analysis." *Journal of Intelligent & Robotic Systems* (2014): 1-16.

Lee, Yong-Cheol, and Charles Eastman. "Formalized Validation Rules and Framework of a Building Information Model Pertaining to a Model View Definition." Paper presented at the *Proceeding of the 32nd CIB W78 Conference* 2015, 2015.

Lee, Yong-Cheol, and Charles Eastman. "Validation Logic for Ensuring the Integrity of Building Information Modeling Data" *Automation in Construction*, Under review (2015).

Lee, Yong-Cheol, Charles M Eastman, and Jin-Kook Lee. "Validations for Ensuring the Interoperability of Data Exchange of a Building Information Model." *Automation in Construction* 58 (2015): 176-95.

Lee, Yong-Cheol, Charles M Eastman, Wawan Solihin, and Richard See. "Modularized Rule-Based Validation of the Industry Foundation Classes Model View Definition." *Automation in Construction*, Accepted in November 2015 (2015).

Lee, Yong-Cheol, Eunhwa Yang, Charles M. Eastman, and Kathy O. Roper. "Modularized Validation of a Building Information Model According to the Specifications of the Facility Management Handover and the Cobie Model View Definition." 2015.

Leite, Fernanda, and Burcu Akinci. "Formalized Representation for Supporting Automated Identification of Critical Assets in Facilities During Emergencies Triggered by Failures in Building Systems." *Journal of Computing In Civil Engineering* 26, no. 4 (2011): 519-29.

McGraw-Hill. "The Business Value of Bim: Getting Building Information Modeling to the Bottom Line." 2009.

Olofsson, Thomas, Ghang Lee, and Charles Eastman. "Editorial-Case Studies of Bim in Use." *IT in Construction* 13 (2008): 244-5.

Sacks, R, I Kaner, C Eastman, and Yeon-Suk Jeong. "The Rosewood Experiment-Building Information Modeling and Interoperability for Architectural Precast Facades." *Automation in Construction* 19 (2010): 419–32.

SECOM, and VIT. "Ifc Model Server Development Project." http://cic.vtt.fi/projects/ifcsvr/download.html#IFCsvr, Retrieved December 13, 2013.

See, R. "Ifc Solutions Factory: Model View Definitions Site." http://www.blis-project.org/IAI-MVD/, Retrieved April 22, 2015.

Solihin, Wawan, Charles Eastman, and Yong-Cheol Lee. "Toward Robust and Quantifiable Automated Ifc Quality Validation." *Advanced Engineering Informatics* 29, no. 3 (2015): 739-56.

Venugopal, M, CM Eastman, and J Teizer. "Formal Specification of the Ifc Concept Structure for Precast Model Exchanges." *Bridges* 10 (2014): 9780784412343.0027.

Weise, M, T Liebich, and J Wix. "Integrating Use Case Definitions for Ifc Developments." eWork and eBusiness in Architecture and Construction. London: Taylor & Francis Group (2009): 637-45.

White, Stephen A. "Introduction to Bpmn." IBM Cooperation 2, no. 0 (2004): 0.

William East, E, Nicholas Nisbet, and Thomas Liebich. "Facility Management Handover Model View." *Journal of Computing In Civil Engineering* 27, no. 1 (2012): 61-67.

Zhang, Chi, Jakob Beetz, and M Weisen. "Interoperable Validation for Ifc Building Models Using Open Standards." *ITcon*, 2015.

VITA

Yong-Cheol Lee is a researcher and a teacher in the area of building information modeling (BIM). During his doctoral studies, he has worked at the Digital Building Laboratory (DBL), led by Professor Charles M. Eastman at the Georgia Institute of Technology. His primary research interests are the interoperability and automated rule-based validation of BIM data. Ensuring the interoperability of BIM data is imperative for improving design communication among architects, engineers, constructors, and their clients. Under the direction of Professor Eastman, he has been involved in research with the Precast/Prestressed Concrete Institute (PCI), the American Institute of Steel Construction (AISC), buildingSMART International, Autodesk Inc., HOK, and Perkins+Will. Under their sponsorship, he has defined the exchange specifications of BIM data and developed automated rule-based validation applications that can guarantee the interoperability and accuracy of BIM data. The goal of such research is to improve design communication using BIM data and facilitate the process of current design and construction work for the architectural, engineering, and construction (AEC) industries through the guaranteed interoperability and correctness of BIM data.

He is also interested in research regarding construction field automation and technology, virtual design and construction, and automated infrastructure and facility asset maintenance using sensors and BIM technology. As a data analytics and research intern at Autodesk, he developed features for integrating city-level BIM and sensing data on top of the InfraWorks platform, which will provide field managers with an efficient work and maintenance environment. While a graduate student in Building Construction at Georgia Tech, he was a graduate research fellow for Professor Minjung Maing, researching improved workflow for the design and fabrication of a building envelope using mechanical CAD tools such as Autodesk Inventor.

During his four years of work experience at the Lotte Engineering & Construction Co., Ltd., he conducted more advanced research pertaining to three-dimensional modeling, BIM coordination, clash detection, site and design management, scheduling, and quality control. His construction industry experience has enabled him to better understand the challenges of construction industry stakeholders and improved his effectiveness at teaching and mentoring future leaders of the industry. To enhance his qualifications, he obtained his Building Engineer's license and LEED AP Building Design + Construction and Facility Management Professional certificates. In addition to his research experience, he has successfully accomplished the responsibilities of a teaching assistant, a guest lecturer, and a student mentor pertaining to BIM, design computation, parametric modeling, construction engineering and management, scheduling, sustainable construction, and construction research. He has received positive feedback from both students and professors regarding his teaching and mentoring abilities. His diverse teaching experience has prepared him to teach a wide range of topics relevant to BIM applications, design computation, sustainable construction, green building, construction management and engineering, construction technology, and construction health and safety.